The RISC Curriculum in Symbolic Computation
July 2005
The RISC Curriculum in Symbolic Computation

The RISC Faculty
faculty@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.uni-linz.ac.at

July 14, 2005

Abstract

This report describes the goals and structure of the curriculum in Symbolic Computation of the Research Institute for Symbolic Computation (RISC) for M.Sc. and Ph.D. students of mathematics or computer science.
Contents

1 RISC and Symbolic Computation  4

2 Studying at RISC  6

3 The RISC Courses  10
   3.1 Computer Algebra  11
   3.2 Computational Logic  12
   3.3 Mathematical Software  13

A Course Descriptions  14
   A.1 Computer Algebra  14
   A.2 Computational Logic  16
   A.3 Mathematical Software  19

B Links to Further Information  22
1 RISC and Symbolic Computation

Symbolic computation is the subarea of mathematics and computer science which solves problems on symbolic objects representable on a computer. Typical examples of such objects are algebraic expressions, logical propositions, and programs themselves. The problem solutions are integrated in many advanced software systems for computer algebra, computer aided design and manufacturing, computer supported reasoning, knowledge management, and formal system specification and verification. Besides playing a fundamental role within mathematics itself, symbolic computation is thus a key technology in many scientific and technical areas today.

Within mathematics itself, the world-wide distribution of computer algebra systems such as Maple or Mathematica has stimulated a variety of new developments towards algorithms and constructiveness. Examples range from classical algebra to analysis, combinatorics, number theory and beyond. The new possibilities to do mathematics with the computer give rise to numerous future challenges for research in symbolic computation. To name only a few examples: the combination of numerical and symbolic scientific computing (e.g., to solve large-scale direct and inverse problems usually described by partial differential equations), the design of automated systems that support the mathematical tasks of proving, solving and simplifying (e.g., the Theorema system at RISC), or the development of methods for computer-assisted mathematical knowledge management (e.g., to make efficient use of mathematical tables like the NIST Digital Library of Mathematical Functions). Symbolic algorithms and their implementations are going to play a more and more significant role not only in natural sciences (biology, chemistry, or physics) but also in industrial engineering. Typical examples are: Gröbner bases in robotics, methods for parametrization of curves and surfaces in geometric modelling and optimization, or software tools in logistics, simulation and telematics.

Extrapolating from these recent developments, one may anticipate that symbolic computation is going to change the paradigm of how mathematical research is done, how mathematics can be taught and applied and how mathematical knowledge is organized, stored, and made accessible. Thus, symbolic computation is not just one particular, special branch of mathematics but a new paradigm that penetrates and changes the entire field of mathematics and its applications. RISC is determined to play a

\[^{1}\text{http://dlmf.nist.gov}\]
leading role in this global paradigm shift and to prepare its PhD students to contribute actively to this development.

More than any other area, symbolic computation depends on the integration of the theoretical foundations (mathematics, logics, algorithms), the implementation in software systems, and the practical applications. It is a main objective of the symbolic computation curriculum at RISC to unite these three aspects. Graduates of this program are not only experts in symbolic computation but, due to their work in highly abstract models with the goal to develop effective software solutions, also professionals in developing innovative computer-based solutions for challenging problems arising in theoretical investigations and real-world applications. Therefore they are highly qualified to find their place in research and development, in academia and industry.

The Research Institute for Symbolic Computation (RISC) (in German: “Institut für Symbolisches Rechnen”) is an institute of the Johannes Kepler University at Linz, Austria. RISC is located in the beautifully renovated medieval castle of Hagenberg, approximately 20km northeast of Linz. The institute was founded by Prof. Bruno Buchberger in 1987 and is currently chaired by Prof. Franz Winkler. In March 2005, Prof. Peter Paule became a new full professor. The entire faculty of RISC consists of 13 members. The working language at RISC is English.

About 25 Ph.D. students (most of them from foreign countries) and a number of M.Sc. students are pursuing their studies at RISC in an inspiring research-oriented working environment. They are integrated into various projects and work in close contact with internationally recognized researchers and with fellow students on their Ph.D. topics at the scientific forefront. They participate in scientific events, publish their results, and thus become part of the international scientific community.

In addition to being the home of Gröbner bases (invented by Bruno Buchberger in 1965), RISC has been the center of numerous scientific, educational, and industrial initiatives, such as the Journal of Symbolic Computation, the RISC Software Company, the Softwarepark Hagenberg, the University of Applied Sciences (Fachhochschule) at Hagenberg, and the Software Competence Center Hagenberg. RISC pursues strong cooperation with the national and international scientific community. It is a key participant in the “Special Research Domain Numerical and Symbolic Scientific Computing” (chaired by Prof. Peter Paule) at the Johannes Kepler University and also helps building up the Johann Radon Institute for Com-
putational and Applied Mathematics (RICAM) of the Austrian Academy of Sciences.

RISC is committed to excellence in research and education. Within the realm of symbolic computation, research at RISC mainly falls into three general categories:

**Computer Algebra** We design and implement algorithms that operate on algebraic expressions; typical application areas are e.g. (algebraic) geometry and (algorithmic) combinatorics.

**Computational Logic** We work on the specification, management, and derivation of knowledge expressed in the language of symbolic logic (resulting in software systems for supporting mathematical proving) and on the theory of computation.

**Mathematical Software** We develop various symbolic computation software such as it occurs in computer algebra systems and theorem provers and study the logical foundations of software for the purpose of formal system specification and verification.

These categories are also reflected in the RISC curriculum. However, we want to emphasize that they mainly serve the clarity of presentation and do not imply an inherent subdivision of the field; rather they present different views on the same subject with strong overlappings and interrelationships (see Figure 1 and the editorial of the Journal of Symbolic Computation\(^2\)). As an example, the Gröbner bases method plays an important role in computer algebra as well as in computational logic and is a core component of many mathematical software systems. Likewise, many courses of the RISC curriculum could be listed under several headers.

## 2 Studying at RISC

The RISC curriculum is open to M.Sc. and Ph.D. students with a background in either mathematics or computer science and a strong interest (and preferably prior knowledge) in the respective other area.

\(^2\)http://www.risc.uni-linz.ac.at/about/editorial
M.Sc. Students of Mathematics or Computer Science  M.Sc. students already hold a bachelor degree in mathematics or computer science and pursue a corresponding master program at the Johannes Kepler University. They can attend RISC courses and get credit for them according to the rules of their program, e.g. as “elective courses”. During their course work, they may get into contact with a RISC professor to agree on the supervision of a master’s thesis and find a suitable thesis topic.

Specifically the master program “Computer mathematics” at the Johannes Kepler University has been designed to allow students of mathematics to specialize in symbolic computation by attending numerous courses of the RISC Symbolic Computation Curriculum. Likewise, the project-oriented elective course groups “Computer Mathematics” and “Formal Methods of Computer Science” allow master students of computer science at the Johannes Kepler University to participate in the symbolic computation curriculum. The details are described in the official curricula of the respective study programs.

Ph.D. Students  Ph.D. students already hold a master degree in mathematics or computer science which entitles them to pursue a doctoral pro-
gram ("Doktorat der technischen Wissenschaften") in Austria in general and at the Johannes Kepler University in particular.

Every year, RISC publishes an international call for applications to Ph.D. studies in Symbolic Computation; details on the application (deadline approximately end of February) can be found on the RISC Web site.

An accepted Ph.D. student subscribes to the doctoral program at the Johannes Kepler University in order to start his or her work at the beginning of the academic year (October). Links to further information on the subscription process can be found in the appendix.

The student is provided with a working place and a computer and gets access to the general RISC infrastructure, i.e. a modern computing environment, state of the art software systems, an up-to-date scientific library, etc. A member of the RISC faculty is assigned as a temporary advisor to assist the student in all academic matters during his or her first year of studies.

A limited number of scholarships is available for students in their first year. Starting from the second year, Ph.D. students are typically employed within the frame of research projects. As part of their general training, students also participate in the setup and administration of the RISC infrastructure (especially the computing environment) and in the organization of scientific events organized by RISC.

The RISC Ph.D. program is equivalent to 210 ECTS points. It takes 3–4 years and is organized as described below.

**Training Semester** The first semester is a probation period in which the student is educated in the basics of the working areas of RISC.

- The student attends the course "Computer Algebra" (4.5 ECTS) which provides a core background for working at RISC.
- The student pursues the course "Thinking, Speaking, Writing" (6 ECTS) which trains mathematical working techniques and communication skills like giving talks and writing papers.
- The student pursues the course "Programming Project Symbolic Computation" (6 ECTS) which trains the basic software development skills.

The student also attends two seminars in order to learn about the ongoing research at RISC (no presentation is required yet).
At the end of the semester, the overall performance of the student with respect to the prospect for a Ph.D. study at RISC is evaluated.

**Course Work** During the second and third semester, the student attends courses amounting to 24 ECTS points (excluding seminars) which corresponds to a total of 8 courses of two hours (the total amount of course work of the Ph.D. curriculum excluding seminars is thus at least 40 ECTS points). The choice of courses is not necessarily restricted to those offered by RISC, but it has to be agreed upon with the advisor at the beginning of every semester.

In the second semester, the student also attends two seminars in order to learn about the ongoing research at RISC (no presentation is required yet).

**Ph.D. Topic** At the end of the second semester, the student agrees with a member of the RISC faculty on the supervision of a Ph.D. thesis, typically in the frame of a research project directed by the faculty. This member of the faculty thus becomes the student’s thesis advisor.

Until the beginning of the second academic year, the student (guided by the advisor) writes a proposal (5–10 pages) for a Ph.D. topic. This proposal is presented to the RISC faculty which decides on the acceptance of the topic or suggests modifications.

**Ph.D. Work** In the subsequent two academic years, the student works on his or her thesis under the guidance of the advisor. The student is expected to take an active part in a seminar reporting there regularly on the progress of the work. At the beginning of the third academic year, the student gives a status report to the RISC faculty.

It is expected that the Ph.D. work results in at least one refereed (conference or journal) publication with the student as the main author.

**Ph.D. Thesis** After three academic years, in accordance with the advisor, the student starts writing the thesis document.

Before submitting the thesis to the university, it is made accessible to the RISC faculty in electronic form; in addition, a corresponding oral presentation is given. The RISC faculty may suggest modifications which are to be incorporated before official submission. The thesis is published in the RISC technical report series.

**Ph.D. Defense** The thesis is presented and defended according to the rules of the Johannes Kepler University.
3 The RISC Courses

RISC offers three kinds of courses:

**Curriculum Courses** These are the “core courses” of the curriculum; every Ph.D. student performs course work in the amount of at least 40 ECTS points. The curriculum courses are offered annually or biennially; additionally “Special Topics” courses with varying content may be offered.

**Seminars** In seminars, the RISC faculty discusses the state of the art in research and ongoing project work. In the first year, every Ph.D. student attends (as a listener) at least two seminars per semester; later the student attends (as an active participant) at least one seminar per semester. Seminar topics vary from semester to semester, the ones listed on the following pages are just examples.

**Bachelor/Master Courses** These are courses offered by RISC for bachelor/master students in mathematics or computer science. They are actually not part of the RISC curriculum but they may help in exceptional cases to supplement the education of new Ph.D. students.

According to the general working areas of RISC, the courses are structured into three categories. Short course descriptions are given in the appendix.
3.1 Computer Algebra

Curriculum Courses

<table>
<thead>
<tr>
<th>Title</th>
<th>ECTS</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Algebra</td>
<td>4.5</td>
<td>annually</td>
</tr>
<tr>
<td>Elimination Theory</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Commutative Algebra and Algebraic Geometry</td>
<td>7.5</td>
<td>biennially</td>
</tr>
<tr>
<td>Algorithmic Algebraic Geometry</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Computer Analysis</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Algorithmic Combinatorics</td>
<td>4.5</td>
<td>annually</td>
</tr>
<tr>
<td>Analytic Combinatorics</td>
<td>3</td>
<td>annually</td>
</tr>
<tr>
<td>Symbolic Linear Algebra</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Special Functions 1</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Special Functions 2</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Symbolic Summation 1</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Symbolic Summation 2</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Category Theory for Symbolic Computation</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Special Topics in Computer Algebra</td>
<td>3</td>
<td>on demand</td>
</tr>
</tbody>
</table>

Seminars (Examples)

- Seminar “Computer Algebra”
- Seminar “Algorithmic Combinatorics”
- Seminar “Algebraic Geometry”

Bachelor/Master Courses

- Algorithmic Methods 1
- Linear Algebra 1 and 2
- Mathematics 1 (Analysis)
3.2 Computational Logic

Curriculum Courses

<table>
<thead>
<tr>
<th>Title</th>
<th>ECTS</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thinking, Speaking, Writing 1+2</td>
<td>6</td>
<td>annually</td>
</tr>
<tr>
<td>Mathematical Logic 1</td>
<td>7.5</td>
<td>annually</td>
</tr>
<tr>
<td>Mathematical Logic 2</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Automated Reasoning 1</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Automated Reasoning 2</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Computability Theory</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Decidability and Complexity Classes</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Decidable Logical Theories</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Rewriting in Computer Science and Logic</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Unification Theory</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Special Topics in Computational Logic</td>
<td>3</td>
<td>on demand</td>
</tr>
</tbody>
</table>

Seminars (Examples)

Seminar “Automated Reasoning (Theorema)”
Seminar “Set Theory and Logical Foundations”

Bachelor/Master Courses

Logic as a Working Language
Mathematical Logic and Logic-Oriented Programming Languages
3.3 Mathematical Software

Curriculum Courses

<table>
<thead>
<tr>
<th>Title</th>
<th>ECTS</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Project Symbolic Computation 1+2</td>
<td>6</td>
<td>annually</td>
</tr>
<tr>
<td>Formal Methods in Software Development</td>
<td>6</td>
<td>annually</td>
</tr>
<tr>
<td>Formal Semantics of Programming Languages</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Formal Specification of Abstract Datatypes</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Formal Models of Parallel a. Distributed Systems</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Design and Analysis of Algorithms</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Introduction to Parallel and Distributed Computing</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Logic Programming</td>
<td>3</td>
<td>annually</td>
</tr>
<tr>
<td>Functional Programming</td>
<td>3</td>
<td>annually</td>
</tr>
<tr>
<td>Programming in Mathematica</td>
<td>3</td>
<td>annually</td>
</tr>
<tr>
<td>Computer Algebra Systems</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Automated Reasoning Systems</td>
<td>3</td>
<td>biennially</td>
</tr>
<tr>
<td>Special Topics in Mathematical Software</td>
<td>3</td>
<td>on demand</td>
</tr>
</tbody>
</table>

Seminars (Examples)

Seminar “Formal Methods”

Bachelor/Master Courses

Practical Software Technology
Software Engineering
Project Engineering
Algorithms and Data Structures
Information Systems
Computer Systems
Formal Foundations 2
Project Practical “Computer Mathematics”
Project Practical “Formal Methods in Computer Science”
A  COURSE DESCRIPTIONS

A  Course Descriptions

In the following, we give short descriptions of the courses of the RISC curriculum.

A.1  Computer Algebra

- **Computer Algebra**: ECTS 4.5, annually.
  A theoretical introduction to the area of computer algebra is presented. The list of topics includes basic algebraic domains, greatest common divisors of polynomials, the Chinese remainder algorithm, factorization of polynomials, and the theory of Gröbner bases.

- **Elimination Theory**: ECTS 3, biennially.
  This course is concerned with algorithmic approaches to the solution of polynomial equations. Typical approaches include resultants, Gröbner bases and characteristic sets.

- **Commutative Algebra and Algebraic Geometry**: ECTS 7.5, biennially.
  In this course the relation between the algebraic theory of polynomial ideals and the geometry of curves, surfaces, and algebraic varieties is described. Topics include Hilbert’s basis theorem, Hilbert’s Nullstellensatz, polynomial and rational functions on varieties, singular points, parametrizations, and determination of dimension.

- **Algorithmic Algebraic Geometry**: ECTS 3, biennially.
  Algebraic Geometry aims at the understanding of the structure of algebraic varieties, which are defined as the solution sets of algebraic equation systems. As this branch of mathematics has a long and rich history, the amount of available theorems is enormous. In Algorithmic AG, we have a look at some of these results from the constructive point of view: instead of being content with the mere existence of some structure (like singularity resolution, special divisors, fibrations), we want to compute it effectively.

- **Computer Analysis**: ECTS 3, biennially.
  Symbolic methods for integration and the analysis and solution of differential equations are described.

- **Algorithmic Combinatorics**: ECTS 3, annually.
  The course provides an introduction to enumerative combinatorics.
Topics are: special sequences like Stirling numbers, partition theory, and important general concepts like group actions and Polya’s counting theory. Theoretical exposition will be supplemented by computer algebra methods.

- **Analytic Combinatorics**: ECTS 3, annually.
  The major topics of this course include sums, recurrences, elementary number theory, binomial coefficients, generating functions, and asymptotic methods. The emphasis is on manipulative technique rather than on existence theorems or combinatorial reasoning. Theoretical exposition will be supplemented by computer algebra methods like Zeilberger’s algorithm. (Literature: “Concrete Mathematics - A Foundation for Computer Science” by R.L. Graham, D.E. Knuth und O. Patashnik, and other books and papers.)

- **Special Functions 1**: ECTS 3, biennially.
  Some functions from analysis arise so often in applications that we give them special names. Paul Turan once remarked that such special functions would be more appropriately labeled “useful functions”. Beginning with Newton and Leibniz, special functions have been continuously developed, including significant contributions by Euler, Legendre, Laplace, Gauss, Kummer, Eisenstein, and Riemann. In recent years the interest in special functions has been renewed, also due to new computer algebra applications. The lecture introduces to fundamental themes like gamma function, hypergeometric series, and Bessel functions. Theoretical exposition will be supplemented by computer algebra methods. (Literature: “Special Functions” by G.E. Andrews, R. Askey, and R. Roy.)

- **Special Functions 2**: ECTS 3, biennially.
  The course is designed in such a way that it can be followed independently from having attended “Special Functions 1”. Topics include: orthogonal polynomials (e.g., Chebyshev, Hermite, Laguerre, Jacobi) and related themes like the irrationality of the Riemann zeta function at $z=3$; in addition, an introduction to q-series is provided. As in Part I, theoretical exposition will be supplemented by computer algebra methods. (Literature: “Special Functions” by G.E. Andrews, R. Askey, and R. Roy.)

- **Symbolic Summation 1**: ECTS 3, biennially.
  The lecture introduces the theoretical foundations of symbolic summation: greatest factorial factorization (GFF) in connection with in-
definite summation of rational and hypergeometric sequences. As applications, famous algorithms as those of Gosper and Zeilberger are discussed.

- **Symbolic Summation 2**: ECTS 3, biennially.  
  Based on works by Karr, the lecture presents a difference field approach to symbolic summation. It can be seen as a discrete analogon to Risch integration theory. Due to work of Schneider, various extensions of Karr’s theory have become a powerful tool which can be used in highly nontrivial applications in special functions, combinatorics, physics, etc.

- **Symbolic Linear Algebra**: ECTS 3, biennially.  
  In many applications of symbolic computation (e.g., summation, integration, solving difference/differential equations) one has to solve systems of linear equations that are not defined over floating-point numbers, but for instance over rational function fields or over principle ideal domains. In this lecture we discuss how to generalize and/or optimize the well-known linear algebra methods in order to solve such systems. The application of these methods are illustrated by various examples.

- **Category Theory for Symbolic Computation**: ECTS 3, biennially.  
  Directional structures between two objects occur widely in mathematics and computer science. Category theory emerged as a frame for formalizing such arrow concepts. Its capacious applicability originates from describing structure in terms of the existence of arrows and relations between them rather than intrinsic properties of objects. Categories themselves arise as abstraction of essentially algebraic theories. Besides its non-constructive parts category theory is concerned with concepts which are mainly algorithmic in nature. In this course we concentrate on this particular aspect of the theory.

- **Special Topics in Computer Algebra**: ECTS 3, on demand.

### A.2 Computational Logic

- **Thinking, Speaking, Writing 1+2**: ECTS 6, annually.

  **TSW 1: Understanding and creating mathematical proofs**  
  RISC is one of the few institutes in the world offering the lecture
“Thinking, Speaking, Writing”, whose purpose is to train the student in all activities involved in scientific work. The first part of the course consists in training in formal reasoning: understanding formal mathematical texts, formalizing mathematical notions, and especially working with formal mathematical texts (proving). As we consider the ability to develop correct proofs as being at the very core of the activity in computer mathematics, the lecture consists in intensive training in basic proof techniques and in-depth understanding and mastering of formal mathematics.

**TSW 2: Communication of scientific results** The career of a scientist relies heavily of his ability to communicate his results through scientific publications and conference talks. The second part of TSW teaches the basic principles and trains the abilities necessary to communication of scientific results in an effective and efficient manner. The subjects include: presentation of scientific results in oral and written form, working with the scientific literature, attending scientific presentations, and cooperating in research groups.

- **Mathematical Logic 1**: ECTS 7.5, annually.
  The course is an introduction to logic for students in Computer Science and Mathematics. The purpose of the course is to understand the principles of Mathematical Logic and its mathematical models, and to acquire the skills for using it in Mathematics and Computer Science. The course covers the main models: propositional logic, first-order predicate logic, higher-order logic, as well as the main proof systems for these models. Furthermore the lecture demonstrates and trains the practical use of Mathematical Logic in Mathematics (building theories, proving), and in Computer Science (automatic reasoning, programming, describing and proving properties of programs).

- **Mathematical Logic 2**: ECTS 3, annually.
  In this course some deeper results in the area of predicate logic are presented, like Gödel’s completeness theorem, omitting types theorem, Craig’s interpolation theorem, Gödel’s incompleteness theorem, as well as applications of elementary chains and ultraproduc ts.

- **Automated Reasoning 1**: ECTS 3, biennially.
  Introduction to Automated Reasoning: This course presents and
compares the basic methods for Automated Reasoning: the resolution method and the main methods for proving in natural style. The topics of the lecture are: syntax and semantics of logical formulae, normalization of formulae, the Herbrand Theorem, the proof method by resolution, the sequent calculus and its use in automated reasoning, basic methods for reasoning with equality. Some concrete implementations will be also presented, including the implementation in Theorema of natural style proving, as well as some applications of automated reasoning in Mathematics and Computer Science.

- **Automated Reasoning 2**: ECTS 3, biennially.
  Advanced Topics in Automated Reasoning: This course presents and compares some advanced methods for automated reasoning: refinements of the resolution method, the use of paramodulation for theories with equality, search strategies in natural style proving, the use of meta-variables in sequent calculus, natural deduction for theories with equality and the Knuth-Bendix completion method, methods for combining general and domain specific reasoning, logic-based algorithm synthesis. The course will also present some practical implementations of these methods, including natural deduction in the Theorema system, as well as current applications of automated reasoning in Mathematics and Computer Science.

- **Computability Theory**: ECTS 3, biennially.
  This course deals not with algorithms for specific problems, but with the notion of computability itself. This means that the class of algorithmically computable (partial recursive) functions is investigated mathematically. Among other things it is shown that certain problems are algorithmically undecidable, that is, they cannot be solved by computer programs even in principle.

- **Decidability and Complexity Classes**: ECTS 3, biennially.
  The first part of this course is a sequel to Computability Theory. Algorithmically undecidable problems may be reduced to each other and thus compared with respect to the degree of undecidability. The second part deals with various aspects of abstract complexity theory, including NP-completeness.

- **Decidable Logical Theories**: ECTS 3, biennially.
  In this course decision algorithms are given for various logical theories, specifically, for the set of all true sentences in some model or the set of all consequences of some set of axioms. Some examples are
Presburger arithmetic, and the theories of real closed fields, Abelian
groups and linear orderings. Various basic methods for developing
such decision algorithms are introduced, the most important being
quantifier elimination.

- **Rewriting in Computer Science and Logic**: ECTS 3, biennially.
  Equational theories appear in many areas of mathematics, logic, and
  computer science. Methods for turning a set of equational axioms
  into an equivalent set of rewrite rules are discussed. The Knuth-
  Bendix procedure and the Gröbner basis algorithm are typical exam-
  ples of such transformations.

- **Unification theory**: ECTS 3, biennially.
  Unification, or equation solving in abstract algebras, has been stud-
  ied since the beginning of mathematics. It is concerned with the
  problem of identifying given terms, either syntactically or modulo
  a given theory. Unification is at the very heart of computation in a
  given logic: It is the basic operation in automated reasoning, rewrit-
  ing, completion, logic programming, and in the related areas. This
course presents first-order syntactic and equational unification, and
higher-order unification and matching.

- **Special Topics in Computational Logic**: ECTS 3, on demand.

### A.3 Mathematical Software

- **Programming Project Symbolic Computation 1+2**: ECTS 6, annu-
  ally.
  In this programming project, the overall process of the team-oriented
development of non-trivial mathematical software using modern de-
velopment tools is exercised.

- **Formal Methods in Software Development**: ECTS 6, annually.
  This course gives a survey on the theory and practice of formal
  methods in the development of sequential and concurrent software,
as they are more and more used in industrial practice for ensuring
the correctness of critical system components. Introducing the log-
ical foundations of of Hoare Calculus, predicate transformers, and
programs as relations on system states, we will investigate various
tools for program specification (e.g. the Java Modeling Language
JML), extended static checking (e.g. Esc/Java2), model checking (e.g.
the model checker SPIN), and program verification by computer-supported proving (e.g. the prototype verification system PVS).

- **Formal Semantics of Programming Languages:** ECTS 3, biennially. This course describes various approaches to define the meaning of a programming language in a mathematically precise way and to reason on this basis about the constructs of the language. These approaches are denotational semantics (programs as functions on semantic domains), operational semantics (programs as state transitions), and axiomatic semantics (programs described by pre/post-condition pairs). We investigate their core ideas, their domain of applicability, and their relationship.

- **Formal Specification of Abstract Datatypes:** ECTS 3, biennially. This course describes the axiomatic/algebraic approach of specifying abstract datatypes (respectively “classes” in object-oriented terminology) in a mathematically precise way. We will introduce the core theory, investigate various application examples, and demonstrate languages and software systems (e.g. CafeOBJ and CASL) for writing, checking, and even executing such specifications.

- **Formal Models of Parallel and Distributed Systems:** ECTS 3, biennially. Concurrent and reactive computing systems are difficult to grasp, because they generally exhibit in different runs different behaviors, due to the time-dependent interleaving of operations from different processes. For an adequate understanding of such systems, formal models are of crucial importance. This course introduces two fundamental approaches to specify the behavior and to reason about the properties of such systems, one based on temporal logic (as exemplified by the Temporal Logic of Actions TLA) and one based on process calculus/algebra (as exemplified by CCS and the pi-Calculus).

- **Design and Analysis of Algorithms:** ECTS 3, biennially. In this course algorithms are presented for specific problems in various areas of mathematics like sorting, graph theory, pattern matching and arithmetic. It also includes the discussion of general principles of algorithm design like ‘Divide and Conquer’ or dynamic programming, as well as techniques for analyzing the complexity of algorithms.

- **Introduction to Parallel and Distributed Computing:** ECTS 3, biennially.
This course gives an introduction to programming parallel and distributed computer systems. We give an overview on parallel computer architectures, discuss the principles of engineering parallel software, and program simple examples in various programming models, e.g. SPMD (single program, multiple data) programming with automatically parallelizing compilers and message passing programming in the MPI (message programming interface) standard.

- **Logic Programming**: ECTS 3, annually.
  Logic programming paradigm has its roots in deduction, and combines it with the computation of values. This course gives an introduction to logic programming, mainly through the Prolog programming language.

- **Functional Programming**: ECTS 3, annually.
  The course is an introduction to functional programming and to the Lisp programming language. The course covers the basic principles of functional programming (recursive definition of data structures and program construction using recursion, tail-recursion and its relation to iterative programs), as well as the main constructs of the Lisp language (constructing and accessing lists, conditional and iterative programming constructs, use of Lisp for symbolic computation).

- **Programming in Mathematica**: ECTS 3, annually.
  This course illustrates the programming language of the well-known symbolic computation system Mathematica. The course is targeted towards students with a strong mathematical background and the exercises always have a mathematical flavor. We concentrate on features of the language, which make programming in Mathematica different from standard procedural programming as taught in the introductory programming courses for students of Mathematics, thus we emphasize pattern matching, rule-based programming style, functional programming style, and graphics programming.

- **Computer Algebra Systems**: ECTS 3, biennially.
  Computer algebra systems are nowadays routinely used in mathematics, science, and engineering for elaborating and solving problems involving symbolic calculations, such as the exact solution of polynomial equations, of indefinite integrals, of sums and recurrences, and much more. Prominent examples of such software are the commercial systems Maple and Mathematica and the open
source systems Axiom, GAP, and Maxima. This course trains the practical use of such systems by a number of application examples.

- **Automated Reasoning Systems**: ECTS 3, biennially.
  Today there exist a number of tools that support formal reasoning in mathematics and computer science, from simple proof checkers to interactive proof assistants, fully automatic decision procedures, and comprehensive theorem proving systems. Examples of such systems are Isabelle/HOL, Coq, ACL2, PVS, and last but not least the Theorema system developed at RISC. This course trains the practical use of such systems by a number of application examples.

- **Special Topics in Mathematical Software**: ECTS 3, on demand.

## B Links to Further Information

- **RISC Ph.D. Program**
  http://www.risc.uni-linz.ac.at/education/phd
  All information concerning the Ph.D. program at RISC including details on the application.

- **RISC Courses**
  http://www.risc.uni-linz.ac.at/education
  The list of courses offered by RISC in the current semester (and links to the courses offered in the past semesters).

- **Symbolic Computation (An Editorial)**

- **The NIST Digital Library of Mathematical Functions**
  http://dlmf.nist.gov

- **Information for International Students**
  http://www.students.jku.edu
  Information for foreign students on studying at the Johannes Kepler University in Linz.

- **Study and Research in Austria**
  http://www.oead.ac.at/_english/austria
  Information for foreign students on studying in Austria by the
Austrian Foreign Exchange Service ÖAD (download the brochure “Studying in Austria”).

- **Universities and Studying**
  
  http://www.bmbwk.gv.at/fremdsprachig/en/univ/un_index.xml
  
  Information for foreign students on studying in Austria by the Austrian Ministry for Education.

- **Studieren in Österreich — Aufnahme von ausländischen Studierenden**
  
  http://www.bmbwk.gv.at/universitaeten/studieren/Aufnahme_von_auslaendisc3502.xml
  
  Information for foreign students on studying in Austria by the Austrian Ministry for Education (in German).

- **Doktoratsstudien an der Technisch/Naturwissenschaftl. Fakultät**
  
  http://dr.tnf.jku.at
  
  The official rules for pursuing a Ph.D. program at the Johannes Kepler University in Linz (in German).

- **Studienrichtung Technische Mathematik**
  
  http://www.numa.uni-linz.ac.at
  
  Description of the bachelor program “Technical Mathematics” and the master program “Computer Mathematics” at the Johannes Kepler University in Linz (in German).

- **Studienplan Informatik**
  
  http://www.informatik.uni-linz.ac.at/Studium/Studienplan
  
  Description of the bachelor and master programs “Computer Science” at the Johannes Kepler University in Linz (in German).