Università
della
Svizzera
italiana

**Faculty
of
Informatics**

Master of Science in Informatics

2017/18

Informatics.

The Faculty of Informatics at Università della Svizzera italiana stands out as a centre of competence in advanced informatics. In a matter of very few years, it has become one of Switzerland's major poles for teaching and research, ranking third after the two Federal Institutes of Technology, Zurich and Lausanne. The Faculty aims to train informatics experts that are interdisciplinary in approach, with abstract thinking and generalization skills, a sound knowledge of Information technologies and their pervasive application domains, as well as project-management and teamwork abilities. The Faculty also offers the unique opportunity to obtain a double Master's degree in collaboration with Politecnico di Milano, and also a joint Master's degree in collaboration with the University of Milano-Bicocca.

Optional joint programme with Politecnico di Milano.

POLITECNICO MILANO 1863

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Awarded Degree
Master of Science in Informatics
Application Deadline
April 30th / June 30th depending on the nationality of the applicant.
Tuition fees per semester
Residents CHF 2'000.– / international CHF 4'000.–
Duration
4 semesters (2 years) - 120 ECTS
Scholarships
Fondazione per le Facoltà di Lugano
10 study grants for Faculty of Informatics, covers first year of tuition, renewable according to grade.
Contacts/information
www.master.inf.usi.ch
studyadvisor@usi.ch

Goals and contents

The Master of Science in Informatics prepares students for current and emerging technologies in computer science by deepening their theoretical knowledge and sharpening their practical skills. The programme is designed for both Bachelor students who wish to complete their education and professionals seeking to refresh their knowledge and sharpen their skills. The Master combines the study of fundamental aspects of computer science with a practical hands-on approach, preparing professionals for successfully pursuing a career in research and development across any application domain.
The Master of Science in Informatics is characterized by a broad offering of topics and subjects that can be freely combined in a learning path tailored to the needs and interests of each student. At USI, students learn how to envision, design, build and optimize complex software intensive systems. They master the ability to develop automated solutions, introduce them in different business and application domains, and predict and assess their positive impact in the real world.
Students experience the need for a rigorous approach to guarantee the quality of their work while following the most appropriate software engineering methodologies, techniques and state-of-the-art tools. Students can benefit from the research excellence of our Faculty of Informatics by getting involved in ongoing research activities as part of their master thesis project, which can be carried out across the entire second year of the Master.

Language

This programme is entirely held in English. Applicants who are not native English speaker or whose first degree was not taught in English, must supply an internationally recognised certificate to demonstrate a C1 level on the Common European Framework of Reference for language learning (CEFR).

Student profile and admission requirements

Bachelor's degree granted by a recognised university in the field of Computer Sciences or related disciplines.
Further information for applicants graduating from a University of Applied Sciences is available online:

www.master.inf.usi.ch/admission

Career opportunities

Informatics is both the infrastructure and the engine of today's society. It plays a key role in industry (pharma, manufacturing of machinery, chemistry, etc.) as well as the service sector (banking, insurance, trade, transport, administration, etc.) in Switzerland. The national training and research institutions have acquired a considerable reputation worldwide, in particular in the field of Information Technology.
Many IT companies, some of them world leaders, have or are planning to have research and development centres in Switzerland. Considering this, graduates in Informatics have excellent opportunities on the job market.
The demand for well-educated specialists in Informatics is very high and is expected to grow even more. Graduates of the Master of Science in Informatics are prepared to become, for example, a business-savvy software designer for the highly competitive software industry of the 21st century, a system engineer with the skills to design, build, integrate, validate and maintain reliable, secure, and large distributed systems. Or be trained to solve complex problems in interdisciplinary areas like graphics and special effects, intelligent search engines, computer vision and face recognition, and robotics.

Contacts

USI Università della Svizzera italiana
Study Advisory Service
+41 58 666 4795
studyadvisor@usi.ch

Study programme

The study programme consists of four semesters full-time study (120 ECTS). Students select 24 ECTS of foundational courses (over the two years) and 66 ECTS of electives based on their interests, plus a substantial Master's thesis (30 ECTS).
A specialisation can be obtained by writing the Master's thesis and taking 18 ECTS of courses in one of the following research areas:

- Computer Systems
- Geometric and Visual Computing
- Information Systems
- Programming Languages
- Theory and Algorithms.

Fall semester	Foundational Courses	Advanced Programming & Design	6.0
		Algorithms & Complexity	6.0
		Distributed Systems	6.0
		High Performance Computing	6.0
		Machine Learning	6.0
	Electives I	Advanced Networking	6.0
		Distributed Algorithms	6.0
		Mobile Computing	6.0
		Numerical Algorithms	3.0
		Software Engineering	6.0
		Software Performance	6.0
		User Experience Design	6.0
	Electives II	Other courses from the Master programmes offered by the Faculty of Informatics	
Spring semester	Foundational Courses	Information Security	6.0
	Electives III	Advanced Computer Architectures	6.0
		Business Process Modeling, Management and Mining	3.0
		Compilers	6.0
		Computer Aided Verification	6.0
		Computer Vision & Pattern Recognition	6.0
		Data Analytics	6.0
		Geometric Algorithms	6.0
		Geometric Deep Learning	3.0
		Geometry Processing	6.0
		Information & Physics	3.0
		Physical Computing	6.0
		Quantum Computing	6.0
		Robotics	6.0
	Electives IV	Other courses from the Master programmes offered by the Faculty of Informatics	
Fourth semester		Master Thesis*	30.0

Please be aware that slight changes in the study programme may occur.

* Master Thesis can be started in 3rd semester.

Fall semester

Foundational Courses

Advanced Programming & Design

This course teaches concepts and methods of object-oriented and concurrent programming that help create complex software systems that are extensible and scalable. It covers principles of object-oriented programming and design, inclusion polymorphism, single and multiple dispatch, parametric polymorphism, design patterns, functional programming, concurrent programming, and aspect-oriented programming. These concepts are explained in the context of the Java programming language.

Algorithms & Complexity

Algorithms are fundamental to computer science. They are the essence of computer programmes and they lie at the core of any software system. This course will cover fundamental techniques for designing efficient computer algorithms, proving their correctness, and analyzing their performance. The contents include greedy algorithms, divide and conquer algorithms, dynamic programming, network flow, NP completeness and computational intractability, approximation algorithms, and randomized algorithms. Techniques on algorithm design and analyisis will be developed by drawing on problems from across many areas of computer science and related fields.

Distributed Systems

Distributed Systems are ubiquitous in modern computer systems. In general, any computing system composed of interconnected autonomous processors is a distributed system. Therefore, understanding how distributed systems are structured is paramount to master modern computer systems. This course is an introduction to distributed systems. It covers basic principles, architectures, and algorithms of distributed systems. The course surveys various aspects of distributed systems, including distributed systems architectures, networking and internetworking, distributed objects and remote invocation, security, distributed file systems, name services, consistency and replication, fault tolerance, and distributed transactions.

High-Performance Computing

Are you interested in using Europe's faster supercomputers (and getting ECTS credit points for doing so)? Would you like to learn how to write programmes for parallel supercomputers, such as a Cray or a cluster of Graphics Processing Units? The course is designed to teach students how to programme parallel computers to efficiently solve challenging problems in science and engineering, where very fast computers are required either to perform complex simulations or to analyze enormous datasets. It covers basic principles, architectures, and algorithms of parallel systems. The course is structured in four parts:

- Foundations of parallel systems;
- Basic parallel algorithm;
- Parallel programming;
- Parallel applications.

Machine Learning

Introductory Master's Course to Intelligent Systems (IS) or Artificial Intelligence (AI), taught by award-winning experts of the Swiss AI Lab IDSIA, and USI. The focus is on Machine Learning (ML). According to Computer World (2009), expertise in ML is the top skill sought by IT employers. Today ML is everywhere: search engines use it to improve answers to queries, email programmes use it to filter spam, banks use it to predict exchange rates and stock markets, doctors use it to recognize tumors, robots use it to localize themselves and obstacles, video games use it to enhance the player's experience, smartphones use it to recognize objects / faces / gestures / voices / music, etc. After the first few lectures of the basic IS course on ML, IS master students will already know how to train self-learning artificial neural networks to recognize images and handwriting better than any other known method. They will rapidly gain familiarity with state-of-the-art algorithms developed at IDSIA and other AI labs.

Electives I

Advanced Networking

This course covers advanced topics in computer networks, with a blend of theoretical and practical topics. On the theoretical side, the syllabus will cover mathematical foundations of networking, including discussions of queuing theory, control theory, information theory, and optimization. On the practical side, the syllabus will cover concepts and designs related to modern network architectures and technologies (e.g., data-center networks, software-defined networks), protocols (e.g., SPDY, HTTP/2, IPSec), and services (e.g., Zookeeper, DHTs). Students will gain hands-on experience with topics discussed in class through a series of exercises using network simulators and emulators.

Distributed Algorithms

Distributed computing systems arise in a wide range of modern applications. This course surveys the foundations of many distributed computing systems, namely, the distributed algorithms that lie at their core. The course provides the basis for designing distributed algorithms and formally reasoning about their correctness. It addresses issues related to what distributed systems can and cannot do (i.e., impossibility results) in certain system models. The course focuses on three aspects of distributed computing: system models, fundamental problems in distributed computing, and application of distributed algorithms. System models include syn-

chronous versus asynchronous systems, communication models, and failure models. Several fundamental problems are covered, including consensus, atomic broadcast, atomic multicast, atomic commit, and data consistency. Applications of distributed algorithms target various forms of replication techniques.

Mobile Computing

Mobile devices such as mobile phones, smart watches, and other wearable devices can interact seamlessly by relying on available communication infrastructures. This course focuses on challenges and opportunities arising from the use of systems of mobile devices or “mobile sensing systems”. Following an overview of applications enabled by mobile sensing systems the focus will be devoted to the most significant technologies, including hardware platforms, programming environments and tools. Relevant aspects related to the design and development of a mobile sensing system, including the handling of sensors, the design of user interfaces, the management of local and remote sensor data storage, privacy and security issues will be investigated and addressed. In order to gain practical hands-on experience, students will learn in the lab sessions how to design, implement, and demonstrate Android-based mobile sensing applications.

Numerical Algorithms

This course is about the key numerical algorithms that you should really want to know about. How do TrueType fonts work? What is the secret of Google's success? Why is JPEG compression so efficient? The answers to these questions are clever numerical algorithms, based on Bézier curves, eigenvalues, and the discrete cosine transformation, respectively. We will be able to understand and discuss them once we have gone through some preliminary basics, including Newton's method for finding roots, polynomial interpolation, direct and iterative methods for solving linear systems of equations, and Gaussian quadrature. This course refreshes your basic math skills in calculus and linear algebra and shows how to utilize them for solving several real-world problems, like the ones mentioned earlier. We also provide references to the history of these solutions, going back to Newton, Leibniz, Euler, Gauss and others.

Software Engineering

Software engineering is the discipline of building software in a methodical way to ensure that the product satisfies its users' needs, is correct (or, more generally, dependable) and maintainable. The course teaches the students how to organize software development projects, how to analyze and specify software requirements, and how to verify software. The course will focus on the use of formal models and methods in software development. 1. Software lifecycle models. Project planning and management. Cost estimation. Standards. Maturity models. 2. Requirements elicitation and specification. 3. Notations and models for formal specification: state machines and Statecharts, Petri nets, declarative descriptions (Alloy). 4. Verification: testing, formal program verification, model checking. The course will be

based on lectures and exercise sessions. The students will also be given assignments, which will be presented and discussed in class.

Software Performance

This course teaches how the various layers of a computer system interact and affect the resulting performance. It performs two cuts down the system stack: one about the 'state' and the other about the 'behavior' of a system. The discussion of 'state' investigates memory usage of applications, leak detection, garbage collection, virtual memory management, and cache performance. The discussion of 'behavior' investigates call graphs, dynamic class loading, shared libraries and dynamic linking, control flow graphs, exception handling, compiler optimizations, and branch prediction. The course uses Java virtual machines and their internal operation as a running example and teaches basic static and dynamic programme analysis techniques. Given the quantitative aspects of performance, the course introduces basic instrumentation and measurement tools, experimentation and evaluation approaches, and data analysis and visualization techniques.

User Experience Design

This class aims at familiarising students with both the theory behind the discipline of Human Computer Interaction (HCI) and the practical process of User eXperience (UX) design. Students not only develop an awareness and appreciation of the crucial implications of good interfaces in terms of overall system performance and user satisfaction, but also learn core skills needed in order to identify user requirements, envision interfaces and processes, and evaluate competing design options. Students will work in small teams of 3-5 to drive a design project from start to finish. Core skills are introduced in hands-on classes, interspersed with lectures and discussions about the underlying theory.

Spring semester

Foundational Courses

Information Security

This class exposes students to the fundamental concepts of cryptography, network security, and computer security. The growing importance of networks and distributed systems, and their use to support safety-critical applications, has made information security a central issue for systems today. The class centers on two main parts: security foundations (which includes security terminology, core cryptograhic principles, and secure protocols) and applied security (which discusses network security, computer security, software security, and web security). Students learn to critically assess the security properties of a system and make informed decisions about implementing secure processes. Most classes feature in-class labs where students are asked to implement a cryptographic primitive or secure protocol, or attack a vulnerable system.

Electives III

Advanced Computer Architectures

The course builds on previous knowledge in basic computer architecture, and visits the major techniques devised to get higher performance from a single processor, and, later on, from multiprocessors. It describes the concepts of pipelined CPUs, cache architecture and optimization, Instruction-Level parallelism (Superscalar and VLIW architectures), Thread-Level parallelism (fine-grained, coarse-grained, simultaneous multithreading), Data-level parallelism (Vector architectures), and shared-memory multi-processing. The course also includes a project where the Sim-plescalar and Watch simulation tools are used to perform design-space exploration, and to understand the tradeoffs that computer architects must consider between performance and cost.

Business Process Modeling, Management and Mining

Business Process Management combines knowledge from information technology and management sciences and applies it to the automation, analysis, monitoring and improvement of operational business processes within large and small organizations. Without well-designed and well-defined process models, to be reliably and efficiently executed, organizations are unable to compete and will not survive in modern globalized marketplaces. This lecture will introduce the students with notations and methodologies for modeling business processes and support their simulation, improvement, mining and execution within process-oriented information systems.

Compilers

This course studies the construction of optimizing compilers, focusing on the "backend" of the compiler: techniques for generating efficient code on modern architectures. We will cover dataflow analysis, programme optimization, and code generation across basic blocks, procedures, and complete programmes. We will look some of the key challenges for modern compilers and runtime systems: optimization of object-oriented languages, dynamic compilation, garbage collection, dependence analysis, and loop transformations. The bulk of your course grade will come from programming assignments that implement programme analysis, intermediate representations, and optimizations.

Computer Aided Verification

This course introduces the students to an approach to validation of hardware and software based on formal analysis of system behaviors. Among the formal methods, model checking enjoys considerable popularity because of its relatively high degree of automation. This approach has been highly effective in the analysis of CPS. The course presents the foundations of model checking starting from the modelling of systems and properties, and then proceeding with the basic algorithms for model checking. Among other things, the distinction between branching time and linear time is discussed, safety and liveness properties are defined, and the use of logics and automata as specifications is discussed. Various logics are introduced, including CTL*, CTL, and LTL. It is shown that model checking for CTL can be reduced to the computation of fixed points of appropriate monotonic functions, and that LTL model checking is based on the translation of the given formula into a Buechi automaton.

Computer Vision & Pattern Recognition

The purpose of the course is to introduce basic problems and notions in image processing, computer vision, and patter recognition though a common geometric framework and present some classical, industry-standard and state-of-the-art methods through this framework. The course uses tools from differential geometry, calculus of variations, and numerical optimization to address problems such as image recovery (denoising, impainting, deconvolution), filtering (adaptive diffusion, bilateral and non-local means filters), 3D structure reconstruction (shape from shading, stereo, photometric stereo); and rigid and non-rigid similarity and correspondence (iterative closest point methods, multidimensional scaling, Gromov-Hausdorff distance). The emphasis is made on both formulating a rigorous mathematical model of the problem and developing an efficient numerical method for its solution, with hands-on programming exercises that solve real-world problems.

Data Analytics

The course deals with mining very large datasets, analysing them to make some descriptive summary of their content, test hypothesis, and extract valuable knowledge from them. Differently from other data mining courses here we deal with datasets that for their large size, speed of updating, or variety of content cannot be mined with standard techniques. Hence we will deal with topics such as: similarity measures for very large datasets and data streams, link analysis, clustering, recommender systems, MapReduce, etc. As part of the course we also learn how to use the R statistical programming language to perform, interpret and visualise results, and diagnose potential problems of your analysis.

Geometric Algorithms

This course is an introduction to computational geometry and its applications. It covers techniques needed in designing and analyzing efficient algorithms for computational problems in discrete geometry such as convex hulls, triangulations, geometric intersections, Voronoi diagrams, Delaunay triangulations, arrangements of lines and hyperplanes, and range searching. Computational geometry is well related to diverse application domains, where geometric algorithms play a fundamental role, such as pattern recognition, image processing, computer graphics, robotics, geographic information systems (GIS), computer-aided design (CAD), information retrieval, computational science, and many others. The course covers general algorithmic techniques, such as plane sweep, divide and conquer, incremental construction, randomization, and approximation, through their application to basic geometric problems.

Geometric Deep Learning

In the past decade, deep learning methods have achieved unprecedented performance on a broad range of problems in various fields from computer vision to speech recognition. However, so far research has mainly focused on developing deep learning methods for Euclidean-structured data. However, many important applications have to deal with non-Euclidean structured data, such as graphs and manifolds. Such geometric data are becoming increasingly important in computer graphics and 3D vision, sensor networks, drug design, biomedicine, recommendation systems, and web applications. The adoption of deep learning in these fields has been lagging behind until recently, primarily since the non-Euclidean nature of objects dealt with makes the very definition of basic operations used in deep networks rather elusive. The purpose of the course is to introduce the emerging field of geometric deep learning on graphs and manifolds, overview existing solutions and applications for this class of problems, as well as key difficulties and future research directions. The course will be held in the form of a seminar; following an introduction by the instructor, the students will present topics related to the field.

Geometry Processing

3D geometry is fundamental to many applications, including virtual characters for movies, interactive design of cars and air planes, and complex simulations. This course covers the whole 3D geometry processing pipeline from scanning real objects to printing them. In the first part we review methods for measuring points on the surface of an object and learn how to align the resulting point clouds. We then discuss how to convert this data into a triangle mesh and study different data structures for handling the latter. The second part explains the main processing tasks for 3D geometry, including smoothing, parameterization, remeshing, decimation, and compression of triangle meshes. In the last part, we talk about 3D printing. We not only cover the relevant theory, but also implement all techniques. For all programming tasks we provide a framework, so that you can concentrate on implementing the core methods and algorithms. The whole pipeline will come to life as you apply it to an object of your choice.

Information & Physics

According to the physicist Rolf Landauer, "information is physical." Starting from this insight, we explore topics at the intersection of physics (thermodynamics, quantum theory, relativity) and information (cryptography, Shannon theory, correlations). Examples - of subjects are the second law of thermodynamics, the arrow of time, the evolution of life and the second law; quantum non-local correlations, causality, the measurement problem and interpretations of quantum theory (collapse models vs. deterministic models such as many-worlds) and the possibility of experimental tests; the relevance of physics for logic, computation, and cryptography, quantum logic, reversible computing; randomness and the emergence of spacetime from information principles. After an introductory part by the teacher, the course is carried out as a seminar, where each participant presents a research paper in a talk. No particular background knowledge is required.

Physical Computing

Physical Computing is about integrating the real world with sensing, communication, and computation. It is about rapidly prototyping devices that can react and interact directly with their environment, rather than being accessed through a keyboard and monitor. The class introduces students to the idea of using small, programmable microcomputers to build self-contained, physical systems that help automate everyday tasks. The course exposes students to basic electronics, microcontroller programming, wireless networking (WiFi and Bluetooth), mobile interfaces (smartphones), and embedded sensing. The class centers on Arduino and ESP development boards that allow one to rapidly build reactive and/or interactive everyday items, without the need for attaching a Mac or PC to them.

Quantum Computing

Followed by an introduction to the basic principles of quantum physics, such as superposition, interference, or entanglement, a variety of subjects are treated: Quantum algorithms, teleportation, quantum communication complexity and "pseudo-telepathy", quantum cryptography, as well as the main concepts of quantum information theory.

Robotics

Robotics addresses the design, construction, and automatic control of mechatronical systems. The course provides a general overview of robotics, focusing on autonomous mobile robots: autonomous systems which exist and move in the physical world, can sense their environment using multiple sensors, can reason about it to issue plans, and can act on it to achieve one or multiple goals. The fundamental concepts and models necessary to achieve such a view of a robotic system will be studied: Forward and Inverse Kinematics; Proprio- and Exteroceptive Sensing; Model-based and Model-free State Estimation; Feedback-based Control; Paradigms and Architectures for Robot Control; Localization and Mapping; Motion Planning; Navigation; Coordination and Cooperation in Multi-robots and Swarms. The course includes theory classes, hands-on classes, and homework. Students will learn how to use the ROS and the simulator Gazebo, and will apply the learned concepts through the programming of both simulated and real robots.

Fourth semester

Master Thesis

The Master thesis is an academic piece of work, an original contribution to the body of knowledge in informatics. Such a contribution can be theoretical or experimental, but always builds on a solid research effort, and on the use of appropriate concepts, methods, and tools acquired during the Master. Faculty members advise students during their Master's thesis work.

Università
della
Svizzera
italiana

**Faculty
of
Informatics**

**Master of Science
in Informatics**

2017/18