SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610656 - Theory of Programming Languages (TLP)
Course in Spanish:	Teoría de lenguajes de programación
Year and term:	1st (Q1)
ECTS:	6
Type:	Compulsory
Teaching language:	English
Module:	Compulsory
Subject:	Fundamental Formal Methods
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	Ignacio Fábregas Alfaro

REQUIRED CONTENTS

- Lambda calculus
- Type systems, sequent calculus, deduction rules
- Abstract reduction machines
- Rewriting
- Semantics of languages: operational and denotational

DETAILED PROGRAM

1. Introduction and review of background.
2. Reduction and rewriting
   Abstract reduction systems
   Term rewriting systems
3. Semantics
   Operational semantics
   Domains and denotational semantics
4. Lambda calculi
   The untyped lambda calculus
   The simply typed lambda calculus
5. Type systems
   Natural deduction and the Curry-Howard isomorphism
   Polymorphism
   Recursion
   Subtyping

DETAILED PROGRAM IN SPANISH

1. Introducción y preliminares
2. Reducción y reescritura
   Sistemas abstractos de reducción
   Sistemas de reescritura
3. Semántica de lenguajes
   Semántica operacional
   Teoría de dominios y semántica denotacional

4. Lambda cálculo
   Lambda cálculo sin tipos
   Lambda cálculo con tipos simples
5. Sistemas de tipos
   Deducción natural e isomorfismo de Curry-Howard
   Polimorfismo
   Recursión
   Subtipado

TRAINING RESULTS

Knowledge:

- RAK1: Compare and evaluate type systems in different programming languages / Comparar y valorar los sistemas de tipos en diferentes lenguajes de programación.
- RAK2: Understand and analyse different formalizations of the semantics of a programming language / Comprender y analizar distintas formalizaciones de la semántica de un lenguaje de programación.
- RAK4: Compare different theoretical models in the study of computer systems / Comparar distintos modelos teóricos en el estudio de sistemas informáticos.
- RAK5: Relate, pointing out strengths and weaknesses, different modeling of computer systems / Relacionar, señalando puntos fuertes y débiles, distintas modelizaciones de sistemas informáticos.

Skills:

- RAS7: Design type systems for programming languages / Diseñar sistemas de tipos para lenguajes de programación.

Competencies:

- RAC18: Define and analyse the formal semantics of a programming language / Definir y analizar la semántica formal de un lenguaje de programación.
- RAC20: Find the appropriate formalization and validation strategy to analyse correctness properties in computer systems / Encontrar una formalización y una estrategia de validación formal adecuada para analizar propiedades de corrección en sistemas informáticos.

LEARNING ACTIVITIES

	Total hours
Master classes	24
Problem solving	21
Laboratory Sessions	0
Oral presentations	0
Taking exams	3
Total	**48 hours**
Independent study	102 hours
Total	**150 hours**

DETAILED EVALUATION

In the regular assessment period the assessment method will consist of two parts [A] and [B]. Students shall hand assignments or projects throughout the course.

[A] Continuous evaluation.

50% - Assignments and small projects.

[B] Exam.

50% - In-class exam.

In the extraordinary assessment sitting, students' grades from "Assignments and small projects" in the ordinary sitting will be kept (method [A] - 50%); the rest of the mark will be graded on the basis of the in-class exam (method [B] - 50%).

BIBLIOGRAPHY

- Franz Baader, Tobias Nipkow. Term rewriting and all that. Cambridge University Press, 1999.
- Henk P. Barendregt. The Lambda Calculus: Its Syntax and Semantics: v.103. Studies in Logic and the Foundations of Mathematics. North Holland, 2014.
- Manuel Clavel et al. All about Maude - A high performance logical framework. Lecture Notes in Computer Science, 4350. Springer, 2007.
- Maude System, version 3.4 https://github.com/maude-lang/Maude/releases/tag/Maude3.4
- Jean-Yves Girard, Yves Lafont, Paul Taylor. Proofs and Types. Cambridge Tracts in Theoretical Computer Science, 7. Cambridge University Press, 1989
- Roger Hindley. Basic Simple Type Theory. Cambridge Tracts in Theoretical Computer Science, 42. Cambridge University Press, 1997.
- Jan van Leeuwen (editor). Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics. The MIT Press, 1991.
- John C. Mitchell. Foundations for Programming Languages. Foundation of computing series. MIT Press, 1996.
- Hanne Riis Nielson, Flemming Nielson. Semantics with Applications: An Appetizer; Undergraduate Topics in Computer Science. Springer, 2007.
- Benjamin Pierce. Types and Programming Languages. MIT Press, 2002.
- Terese. Term Rewriting Systems. Cambridge Tracts in Theoretical Computer Science, 55. Cambridge University Press, 2003.
- Glynn Winskel. The Formal Semantics of Programming Languages: An Introduction, 4th. ed.; Foundations of Computing. MIT Press, 1997.

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Curso 2025-2026

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610608 - Static analysis of programs and constraint solving (AERR)
Course in Spanish:	Análisis Estático y Resolución de Restricciones
Year and term:	1st (Q1)
ECTS:	6
Type:	Compulsory
Teaching language:	English
Module:	Compulsory
Subject:	Fundamental Formal Methods
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	María Elena Gómez Martínez

REQUIRED CONTENTS

- Axiomatic semantics
- Abstract interpretation
- Abstract domains
- Analyses based on types
- SAT and SMT solvers
- Constraint programming
- Satisfaction and optimization problems
- Constraint solvers

DETAILED PROGRAM

1. Introduction to static program analysis.
2. Type-based analysis: typed arithmetic expressions and typed object-oriented programs, references, structural and nominal subtyping. Type safety, type and effect systems.
3. Abstract interpretation: Galois connections, fixed point theory, widening and narrowing.
4. Abstract domains: sign analysis, intervals, polyhedral domains.
5. SAT and SMT solvers: boolean constraint propagation, DPLL(T), equality logic, uninterpreted functions, array theory.
6. Constraint programming.
7. Satisfaction and optimization problems.
8. Constraint solvers.

DETAILED PROGRAM IN SPANISH

1. Introducción al análisis estático de programas.
2. Análisis basados en tipos: tipado de expresiones aritméticas, programas orientados a objetos, referencias, subtipado estructural y nominal, seguridad de tipos, sistemas de tipos y efectos.
3. Interpretación abstracta: conexiones de Galois, teoría de puntos fijos, operadores de ensanchamiento y estrechamiento.
4. Dominios abstractos: análisis de signo, intervalos y dominios poliédricos.
5. Resolutores SAT y SMT: propagación de restricciones booleanas, DPLL(T), lógica con igualdad, funciones no interpretadas, teoría de arrays.
6. Programación con restricciones.
7. Problemas de satisfactibilidad y optimización.

8. Resolución de restricciones.

TRAINING RESULTS

Knowledge:

- RAK3: Identify a suitable formalization, maybe using an over approximation, for the analysis of correctness properties in a computer system / Identificar una formalización adecuada, que puede requerir una sobreaproximación, para el análisis de propiedades de corrección en un sistema informático.
- RAK4: Compare different theoretical models in the study of computer systems / Comparar distintos modelos teóricos en el estudio de sistemas informáticos.
- RAK5: Relate, pointing out strengths and weaknesses, different modeling of computer systems / Relacionar, señalando puntos fuertes y débiles, distintas modelizaciones de sistemas informáticos.
- RAK6: Model static analysis problems as mathematical constraint solving problems / Modelar problemas de análisis estático como problemas de resolución de restricciones matemáticas.

Skills:

- RAS8: Design static analyses based on formal methods to analyse properties of computer systems / Diseñar análisis estáticos basados en métodos formales para analizar propiedades de sistemas informáticos.

Competencies:

- RAC18: Define and analyse the formal semantics of a programming language / Definir y analizar la semántica formal de un lenguaje de programación.
- RAC19: Design static analyses of computer programs based on abstract interpretation, type systems and constraint resolution / Diseñar análisis estáticos de programas informáticos basados en interpretación abstracta, sistemas de tipos y resolución de restricciones.
- RAC20: Find the appropriate formalization and validation strategy to analyse correctness properties in computer systems / Encontrar una formalización y una estrategia de validación formal adecuada para analizar propiedades de corrección en sistemas informáticos.

LEARNING ACTIVITIES

	Total hours
Master classes	24
Problem solving	5
Laboratory Sessions	12
Oral presentations	4
Taking exams	3
Total	**48 hours**
Independent study	102 hours
Total	**150 hours**

DETAILED EVALUATION

Students will be graded on the basis of class assignments that will be proposed throughout the course and an oral presentation at the end. The final grade will be computed as follows:

- [35%] Assignments on static analysis and SAT/SMT solvers. Students might be required to present their work.

- [35%] Assignments on constraint solving and optimization. Students are expected to express a given problem as a set of constraints, apply the necessary tools for solving it, and assess the results. Students might be required to present their work.
- [30%] An exam at the end of the course, in which students are required to give an oral presentation on type systems.

In the resit, a new deadline will be set for those students that obtained a failing grade in the regular assessment period. Students are allowed to (re)submit any assignments that failed to obtain a passing grade, and can give the presentation on type systems if they failed to do so in the ordinary sitting.

BIBLIOGRAPHY

• Benjamin C. Pierce. Types and Programming Languages. Cambridge, Mass.; MIT Press, 2002.
• Xavier Rival, Kwangkeun Yi. Introduction to static analysis: an abstract interpretation perspective. The MIT Press, 2020
• Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. Principles of Program Analysis. Berlin: Springer, 2005.
• Daniel Kroenig, Ofer Strichman. Decision Procedures: An Algorithmic Point of View (2nd ed.). Texts in Theoretical Computer Science. Springer, 2016.
• Richard L. Ford, K. Rustan M. Leino. Dafny Reference Manual. 2017.
• Francesca Rossi, Peter van Beek, Toby Walsh. Handbook of Constraint Programming. Elsevier Science, 2006.
• Krzysztof Apt. Principles of Constraint Programming. Cambridge Press, 2003.
• Kimbal Marriott, Peter Stuckey. Programming with Constraints: An Introduction. MIT Press, 1998.
• Slim Abdennadher, Thom Frühwirth. Essentials of Constraint Programming. Springer, 2003.

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Curso 2025-2026

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610614 - Concurrency Models (MC)
Course in Spanish:	Modelos de Concurrencia
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	Rigorous systems design and construction
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	Fernando Rosa Velardo

REQUIRED CONTENTS

- Process algebras
- Bisimulation relations
- Temporal logic
- Model checking
- Petri nets

DETAILED PROGRAM

- Process algebras
  - Operational, denotational and axiomatic semantics
  - Bisimulation. Semantic equivalences and orders
  - Logics: specifying and checking properties of systems
- Introduction to modelling, analysis and verification tools (model checking): Mcrl2, Concurrency WorkBench (CAAL)
- Petri nets
  - Classes of nets
  - Basic properties and analysis techniques
  - Petri net tools for modelling and analysis of concurrent systems

DETAILED PROGRAM IN SPANISH

- Álgebra de procesos.
  - Semánticas operacional, denotacional, axiomática.
  - Bisimulación. Equivalencias semánticas y órdenes.
  - Lógicas
- Herramientas para la modelización, análisis y verificación de propiedades (model checking): Mcrl2, Concurrency WorkBench (CAAL)
- Redes de Petri
  - Clases de redes.
  - Propiedades básicas y técnicas para su análisis
  - Introducción a las herramientas basadas en redes de Petri para la modelización y análisis de sistemas concurrentes

TRAINING RESULTS

Knowledge:

- RAK3: Identify a suitable formalization, maybe using an over approximation, for the analysis of correctness properties in a computer system / Identificar una formalización adecuada, que puede requerir una sobreaproximación, para el análisis de propiedades de corrección en un sistema informático.
- RAK4: Compare different theoretical models in the study of computer systems / Comparar distintos modelos teóricos en el estudio de sistemas informáticos.
- RAK5: Relate, pointing out strengths and weaknesses, different modeling of computer systems / Relacionar, señalando puntos fuertes y débiles, distintas modelizaciones de sistemas informáticos.

Skills:

- RAS9: Formally prove or disprove correctness properties of computer systems / Demostrar o desmentir formalmente propiedades de corrección de sistemas informáticos.
- RAS12: Analyze properties of concurrent systems using theoretical models and tools based on them / Analizar propiedades de los sistemas concurrentes utilizando modelos teóricos y herramientas basadas en éstos.

Competencies:

- RAC20: Find the appropriate formalization and validation strategy to analyse correctness properties in computer systems / Encontrar una formalización y una estrategia de validación formal adecuada para analizar propiedades de corrección en sistemas informáticos.
- RAC22: Compare and decide the concurrency model that is best suited to capture the intrinsic characteristics of a concurrent computing system / Comparar y decidir el modelo de concurrencia que mejor se adapte para capturar las características intrínsecas de un sistema informático concurrente.

LEARNING ACTIVITIES

	Total hours
Master classes	34
Problem solving	9
Laboratory Sessions	4
Oral presentations	0
Taking exams	1
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

Regular examination:
There are two tracks of evaluation:
[A] Assignments 80%; Participation during the course 20%
Those students failing evaluation track [A] can follow track [B]:
[B] Assignments 40%; Exam 40%; Participation during the course 20%

Extraordinary examination: Exam 40%; Assignments 40%; Participation during the course 20%.

In the extraordinary assessment sitting, students may be required to submit new assignments before the exam, for which a new deadline will be set. The mark corresponding to participation in the extraordinary examination will be equal to the mark obtained for the regular examination.

BIBLIOGRAPHY

L. Aceto, A. Ingólfsdóttir, K. Larsen, and J. Srba. Reactive Systems: Modelling, Specification and Verification. Cambridge University Press, 2007.

J.F. Groote and M.R. Mousavi. Modeling and analysis of communicating systems. The MIT press. 2014. J. Andersen et al. CAAL: Concurrency Workbench, Aalborg Edition, Theoretical Aspects of Computing - ICTAC 2015, pp. 573--582

J. Desel, W. Reisig, and G. Rozenberg (Eds.). Advances in Petri Nets, Lecture Notes in Computer Science, vol. 3098, Springer-Verlag, 2004.

W. Reisig. Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer 2013.

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610610 - Program-assisted verification (VAP)
Course in Spanish:	Verificación Asistida de Programas
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	System correctness analysis
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	Clara María Segura Díaz

REQUIRED CONTENTS

- Proof assistants
- Analysis of program termination
- Assisted platforms for verification
- Inference systems for qualified types
- Case studies

DETAILED PROGRAM

1. The Dafny verification platform:
   - 1.1. The basics of the Dafny language
   - 1.2. Assisted proofs of termination and lemmas
   - 1.3. Verification of programs on arrays
   - 1.4. Algebraic data types and data structures
   - 1.5. Dynamic frames
2. Programming with refinement types: Liquid Haskell
   - 2.1. Refinement types.
   - 2.2. Measures and data types invariants. Case studies.
   - 2.3. Abstract refinement types.
   - 2.4. Theorem proving.
   - 2.5. Inductive data types.

DETAILED PROGRAM IN SPANISH

1. La plataforma Dafny de verificación asistida.
   - 1.1. El lenguaje de programación Dafny.
   - 1.2. Asistencia a las demostraciones de terminación y de lemas.
   - 1.3. Verificación de programas con vectores.
   - 1.4. Tipos algebraicos y estructuras de datos.
   - 1.5. Marcos dinámicos.
2. Programación con tipos refinados: Liquid Haskell.
   - 2.1. Tipos refinados.
   - 2.2. Medidas e invariantes de tipos de datos. Casos de estudio.
   - 2.3. Tipos con refinamientos abstractos.

2.4. Demostración de teoremas.
2.5. Tipos de datos inductivos.

TRAINING RESULTS

Knowledge:

- RAK3: Identify a suitable formalization, maybe using an over approximation, for the analysis of correctness properties in a computer system / Identificar una formalización adecuada, que puede requerir una sobreaproximación, para el análisis de propiedades de corrección en un sistema informático.
- RAK5: Relate, pointing out strengths and weaknesses, different modeling of computer systems / Relacionar, señalando puntos fuertes y débiles, distintas modelizaciones de sistemas informáticos.
- RAK6: Model static analysis problems as mathematical constraint solving problems / Modelar problemas de análisis estático como problemas de resolución de restricciones matemáticas.

Skills:

- RAS9: Formally prove or disprove correctness properties of computer systems / Demostrar o desmentir formalmente propiedades de corrección de sistemas informáticos.
- RAS10: Specify and verify program properties using assisted demonstration tools / Especificar y verificar propiedades de programas utilizando herramientas de demostración asistida.

Competencies:

- RAC20: Find the appropriate formalization and validation strategy to analyse correctness properties in computer systems / Encontrar una formalización y una estrategia de validación formal adecuada para analizar propiedades de corrección en sistemas informáticos.

LEARNING ACTIVITIES

	Total hours
Master classes	22
Problem solving	11
Laboratory Sessions	12
Oral presentations	0
Taking exams	3
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

- During the course, the student must prepare practical or/and theoretical assignments. The mark obtained here will be kept up to the extraordinary examination and there will not be extra assignments after the ordinary examination. (50%)
- There will be an ordinary examination and an extraordinary one (50%). It is necessary to obtain at least 4 points over 10 in the exam in order to pass. If less than 4 points are obtained in the exam, the final mark will be the minimum between the mark obtained from the exam and the assignments and 4.

BIBLIOGRAPHY

- K. Rustan M. Leino. Program proofs. MIT Press, 2023.
- K. Rustan M. Leino. Dafny: An Automatic Program Verifier for Functional Correctness. In LPAR-16, volume 6355 of LNCS, pages 348-370. Springer, 2010.
- K. Rustan M. Leino. Specification and verification of object-oriented software. In Engineering Methods and Tools for Software Safety and Security, volume 22 of NATO Science for Peace and Security Series D: Information and Communication Security, pages 231-266. IOS Press, 2009
- Online Dafny tutorial, https://dafny.org/dafny/OnlineTutorial/guide
- Dafny reference manual, https://github.com/dafny-lang/dafny/blob/master/docs/DafnyRef/out/DafnyRef.pdf, 2022. Online version (https://dafny.org/dafny/DafnyRef/DafnyRef)
- P. M. Rondon, M. Kawaguchi, R. Jhala. Liquid types. In Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI'08, pages 159-16, 2008.
- M. Kawaguchi, P. M. Rondon, R. Jhala. Type-based data structure verification. In Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI'09.pages 304-315.
- N. Vazou, E.L. Seidel, R. Jhala, S. Peyton-Jones. Refinement Types for Haskell. In Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming, ICFP'14, pages 269-282, 2014.
- N. Vazou, E.L. Seidel, R. Jhala. LiquidHaskell: experience with refinement types in the real world. In Proceedings of the 2014 ACM SIGPLAN Symposium on Haskell, Haskell'14, pages 39-51, 2014.
- N. Vazou, E.L. Seidel, R. Jhala.Abstract refinement types. In 22nd European conference on Programming Languages and Systems, ESOP'13, pages 209-228, 2013.
- Online Liquid Haskell tutorial, http://ucsd-progsys.github.io/lh-workshop/.
- Online Proving Theorems in Lean, https://leanprover.github.io/theorem_proving_in_lean/index.html

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Curso 2025-2026

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610609 - Formal methods in testing (MFT)
Course in Spanish:	Métodos Formales de Testing
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	System correctness analysis
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	José ignacio Requeno Jarabo

REQUIRED CONTENTS

- Formal testing of finite state machines
- Conformance relations
- Combining testing and model checking
- Formal testing of distributed and cloud environments
- Tools for formal testing

DETAILED PROGRAM

1. Introduction to software testing.
2. Introduction to formal methods in testing.
3. Testing from state-based systems.
4. Testing distributed and asynchronous systems.
5. Runtime verification.
6. Model-based testing.

DETAILED PROGRAM IN SPANISH

1. Introducción al testing de software.
2. Introducción a los métodos formales de testing.
3. Testing de sistemas basados en estados.
4. Testing de sistemas distribuidos y asíncronos.
5. Verificación de sistemas en tiempo de ejecución.
6. Testing basado en modelos.

TRAINING RESULTS

Knowledge:

- RAK3: Identify a suitable formalization, maybe using an over approximation, for the analysis of correctness properties in a computer system / Identificar una formalización adecuada, que puede requerir una sobreaproximación, para el análisis de propiedades de corrección en un sistema informático.

- RAK5: Relate, pointing out strengths and weaknesses, different modeling of computer systems / Relacionar, señalando puntos fuertes y débiles, distintas modelizaciones de sistemas informáticos.

Skills:

- RAS9: Formally prove or disprove correctness properties of computer systems / Demostrar o desmentir formalmente propiedades de corrección de sistemas informáticos.
- RAS11: Design liveness and security analyses of concurrent and distributed computing systems / Diseñar análisis de vivacidad y seguridad de sistemas informáticos concurrentes y distribuidos.

Competencies:

- RAC20: Find the appropriate formalization and validation strategy to analyse correctness properties in computer systems / Encontrar una formalización y una estrategia de validación formal adecuada para analizar propiedades de corrección en sistemas informáticos.

LEARNING ACTIVITIES

	Total hours
Master classes	41
Problem solving	0
Laboratory Sessions	0
Oral presentations	5
Taking exams	2
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

Students have to present a certain number of research papers. The number of research papers will be fixed with enough time and it will be announced both during the lectures and in the virtual campus. They will receive a mark between 0 and 10 points. If a student does not present the paper on the assigned date, then the mark corresponding to this presentation, both in the ordinary and extraordinary examination, is equal to zero. All the students must attend the lectures when other students are presenting a paper.
Students will be encouraged to participate in the regular lectures: answering questions and assignments posed by the lecturers.

- Regular examination: Presentations 90%; Participation during the lectures 10%.
- Extraordinary examination: Exam 50%; Presentations 40%; Participation during the lectures 10%.

The marks corresponding to presentations and participation in the extraordinary examination will be equal to the marks obtained for the regular examination.

BIBLIOGRAPHY

- Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, Alexander Pretschner: Model-Based Testing of Reactive Systems. Lecture Notes in Computer Science 3472, Springer 2005.
- Yliès Falcone, Klaus Havelund, Giles Reger: A Tutorial on Runtime Verification. Engineering Dependable Software Systems 2013: 141-175
- Yliès Falcone, Srdan Krstic, Giles Reger, Dmitriy Traytel: A taxonomy for classifying runtime verification tools. Int. J. Softw. Tools Technol. Transf. 23(2): 255-284 (2021)
- Robert M. Hierons, Jonathan P. Bowen, Mark Harman: Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers. Lecture Notes in Computer Science 4949, Springer 2008.
- David Lee, Mihalis Yannakakis. Principles and methods of testing finite state machines - a survey. Proceedings of the IEEE 84 (8), 1090-1123, 1996.

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Curso 2025-2026

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610609 - Analysis of concurrent and distributed systems (ASCD)
Course in Spanish:	Análisis de Sistemas Concurrentes y Distribuidos
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	System correctness analysis
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	Albert Rubio Gimeno

REQUIRED CONTENTS

- Semantics of concurrent and distributed systems
- Basic properties: termination and resource consumption
- Liveness properties: absence of locks and starvation
- Verification based on static analysis
- Validation based on testing
- Implementation and existing tools

DETAILED PROGRAM

1. Semantics of Concurrent and Distributed Programs
2. Dynamic analysis of concurrent and distributed systems
3. Static analysis of concurrent and distributed systems
   - 3.1. Basic and liveness properties
   - 3.2. Termination and resource consumption
4. Analysis and verification of smart contracts
5. Property-based testing of concurrent systems

DETAILED PROGRAM IN SPANISH

1. Semántica de programas concurrentes y distribuidos
2. Análisis dinámico de sistemas concurrentes y distribuidos
3. Análisis estático de sistemas concurrentes y distribuidos
   - 3.1. Propiedad básicas y de vitalidad
   - 3.2. Terminación y consumo de recursos
4. Análisis y verificación de contratos inteligentes
5. Pruebas de sistemas concurrentes basadas en propiedades

TRAINING RESULTS

Knowledge:

- RAK3: Identificar una formalización adecuada, que puede requerir una sobreaproximación, para el análisis de propiedades de corrección en un sistema informático / Identify a suitable formalization, maybe using an over approximation, for the analysis of correctness properties in a computer system.

- RAK5: Relacionar, señalando puntos fuertes y débiles, distintas modelizaciones de sistemas informáticos / Relate, pointing out strengths and weaknesses, different modeling of computer systems.

Skills:

- RAS9: Demostrar o desmentir formalmente propiedades de corrección de sistemas informáticos / Formally prove or disprove correctness properties of computer systems.
- RAS11: Diseñar análisis de vivacidad y seguridad de sistemas informáticos concurrentes y distribuidos / Design liveness and security analyses of concurrent and distributed computing systems.

Competencies:

- RAC20: Encontrar una formalización y una estrategia de validación formal adecuada para analizar propiedades de corrección en sistemas informáticos / Find the appropriate formalization and validation strategy to analyse correctness properties in computer systems.

LEARNING ACTIVITIES

	Total hours
Master classes	33
Problem solving	7
Laboratory Sessions	4
Oral presentations	3
Taking exams	1
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

Evaluation of course exercises (5 groups of exercices, one per topic): 80%
Final exam with a public presentation of a research paper: 20%
The minimum grade required for practice course exercises in each group is 3 out of 10.

The referred evaluation follows the same method but all course exercises can be resubmitted and a new public presentation can be made.

BIBLIOGRAPHY

- Gregory R. Andrews. 1991. Concurrent Programming: Principles and Practice. Benjamin-Cummings Publ. Co., Inc., Redwood City, CA, USA.
- M. Ben-Ari. 1990. Principles of Concurrent and Distributed Programming. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Cormac Flanagan, Patrice Godefroid: Dynamic partial-order reduction for model checking software. POPL 2005: 110-121.
- Krzysztof R. Apt, Frank S. de Boer, Ernst-Rüdiger Olderog: Verification of Sequential and Concurrent Programs. Texts in Computer Science, Springer 2009, ISBN 978-1-84882-744-8, pp. i-xxiii, 1-502.

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610612 - Design of correct-by-construction systems (DSCC)
Course in Spanish:	Diseño de sistemas correctos por construcción
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	Rigorous systems design and construction
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	Albert Rubio Gimeno (at UCM)

REQUIRED CONTENTS

- Impact of software problems in our society.
- Termination, correctness and completeness. Reactive systems. Discrete systems.
- Specification,first-order logic, proofs, sequent calculus.
- Tools.
- Sequential functional systems.
- Reactive systems.
- Data structures.
- Distributed Systems.

DETAILED PROGRAM

1. Introduction: Proving Programs Correct
2. Fundamentals: Specification, First-Order Logic, Proofs, Programs
3. Event-B Basics and the Rodin Tool
4. Sequential Systems
5. Event B: Mathematical Toolkit and Applications
6. Reactive Systems: Concurrency and Distribution

DETAILED PROGRAM IN SPANISH

1. Introducción: Demostración de la corrección de programas
2. Fundamentos: Especificación, Lógica de Primer Orden, Demostraciones, Programas
3. Event-B: Conceptos básicos y la herramienta Rodin
4. Sistemas secuenciales
5. Event B: Conjunto de herramientas matemáticas y sus aplicaciones
6. Sistemas reactivos: Concurrencia y Distribución

TRAINING RESULTS

Knowledge:

- RAK3: Identificar una formalización adecuada, que puede requerir una sobreaproximación, para el análisis de propiedades de corrección en un sistema informático / Identify a suitable formalization, maybe using an over approximation, for the analysis of correctness properties in a computer system.
- RAK5: Relacionar, señalando puntos fuertes y débiles, distintas modelizaciones de sistemas informáticos / Relate, pointing out strengths and weaknesses, different modeling of computer systems.

Skills:

- RAS9: Demostrar o desmentir formalmente propiedades de corrección de sistemas informáticos / Formally prove or disprove correctness properties of computer systems.
- RAS12: Analizar propiedades de los sistemas concurrentes utilizando modelos teóricos y herramientas basadas en éstos / Analyze properties of concurrent systems using theoretical models and tools based on them.
- RAS14: Diseñar y valorar variaciones de un sistema informático existente para que sea posible comprobar propiedades relevantes de manera más simple / Design and evaluate variations of an existing computer system so that relevant properties can be easier to check.

Competencies:

- RAC20: Encontrar una formalización y una estrategia de validación formal adecuada para analizar propiedades de corrección en sistemas informáticos / Find the appropriate formalization and validation strategy to analyse correctness properties in computer systems.
- RAC22: Comparar y decidir el modelo de concurrencia que mejor se adapte para capturar las características intrínsecas de un sistema informático concurrente / Compare and decide the concurrency model that is best suited to capture the intrinsic characteristics of a concurrent computing system.

LEARNING ACTIVITIES

	Total hours
Master classes	29
Problem solving	6
Laboratory Sessions	8
Oral presentations	4
Taking exams	1
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

The assessment method will consist of two parts:
- Assignments to be developed during the course period at home [60%].
- Term project [40%].
The students have the possibility to skip the previous assessment and take only an in-class final exam [100%].

Referred (re-sit) assessment: in-class exam [100%]

BIBLIOGRAPHY

- Lawrence Paulson. Logic and Proofs, class notes. Computer Laboratory, University of Cambridge.
- Michael Huth and Mark Ryan. Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press.
- Jean-Raymond Abrial. Modeling in Event-B: System and Software Engineering. Cambridge University Press.
- http://wiki.event-b.org/

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610615 - Design of bio-inspired algorithm (DABI)
Course in Spanish:	Diseño de algoritmos bioinspirados
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	Specialized techniques in system design
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	Fernando Rubio Diez

REQUIRED CONTENTS

- Introduction to bio-inspired computing models
- Classical models: cellular automata, parallel derivation grammars, DNA-inspired systems
- Evolutionary computation, from genetic algorithms to genetic programming using genetic programming with complex, formal representations of populations
- Other computing models: membrane-based, networks of processors that evolve, etc.

DETAILED PROGRAM

1. Complexity and approximability
2. Bio-inspiration for models of computing
3. Evolutionary computation
   - 3.1. Introduction to genetic algorithms and evolutionary computation
   - 3.2. Combining genetic algorithms with greedy methods
4. Swarm Intelligence
   - 4.1. Introduction to swarm intelligence models and methods
   - 4.2. Ant Colony Optimization and Particle Swarm Optimization

DETAILED PROGRAM IN SPANISH

1. Complejidad y aproximabilidad
2. Bio-inspiración para modelos de cómputo
3. Computación evolutiva
   - 3.1. Introducción a los algoritmos genéticos y computación evolutiva
   - 3.2. Combinación de algoritmos genéticos y voraces
4. Inteligencia de enjambre
   - 4.1. Introducción a los modelos y métodos de inteligencia de enjambre
   - 4.2. Optimización basada en colonias de hormigas y enjambres de partículas

TRAINING RESULTS

Skills:

- RAS15: Compare various bio-inspired computing models for solving hard computational problems / Aplicar diversos modelos de cómputo bioinspirados para resolver problemas computacionales difíciles..

Competencies:

- RAC23: Design bio-inspired algorithms taking into account their performance and suitability to the problem to be solved / Diseñar algoritmos bioinspirados teniendo en cuenta su rendimiento y adecuación al problema a resolver.

LEARNING ACTIVITIES

	Total hours
Master classes	33
Problem solving	6
Laboratory Sessions	8
Oral presentations	0
Taking exams	1
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

The assessment method will consist of three parts:

- Practical work done during the course to understand the basic models: 50%
- Oral presentation of a final project demonstrating fluency and maturity in handling the models: 40%
- Written exam: 10%

BIBLIOGRAPHY

[1] Andries P. Engelbrecht. Computational Intelligence: An Introduction. Willey.
[2] Marco Dorigo and Thomas Stützle. Ant Colony Optimization. The MIT Press.
[3] Riccardo Poli, James Kennedy, Tim Blackwell. Particle Swarm Optimization. Swarm Intelligence, Vol 1, Issue 1, pp. 33-57, 2007.
[4] A. E. Eiben, J. E. Smith: Introduction to Evolutionary Computing, Springer, 2003.
[5] David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989
[6] T. Bäck, D. B. Fogel, Z. Michalewicz. Evolutionary Computation 1: Basic Algorithms and Operators. IoP, 2000.
[7] T. Bäck, D. B. Fogel, Z. Michalewicz. Evolutionary Computation 2: Advanced Algorithms and Operators. IoP, 2000.
[8] Approximation Algorithms. Vijay V. Vazirani. Springer. 2001
[9] Computational Complexity: A Modern Approach. Sanjeev Arora and Boaz Barak. Cambridge University Press. 2009

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Curso 2025-2026

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610616 - Quantum Computing (CC)
Course in Spanish:	Computación cuántica
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	Specialized techniques in system design
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	David Pérez García

REQUIRED CONTENTS

- Quantum information foundations
- Quantum computing
- Shor and Grover algorithms
- Quantum cryptography
- BB84 algorithm

DETAILED PROGRAM

- Fundamentals of quantum information: postulates of quantum mechanics and notation.
- Quantum Computing: Quantum Circuits and Algorithms
- Some classical algorithms in quantum computation
- Quantum cryptography: Uncertainty principle, no-cloning theorem.
- Some basic protocols in quantum cryptography

DETAILED PROGRAM IN SPANISH

- Fundamentos de información cuántica: postulados de la mecánica cuántica y notación.
- Computación cuántica: Circuitos cuánticos y algoritmos
- Algunos algoritmos clásicos en computación cuántica
- Criptografía cuántica: Principio de incertidumbre, teorema de no clonación.
- Algunos protocolos básicos en criptografía cuántica

TRAINING RESULTS

Skills:

- RAS16: Analyse known quantum computing algorithms and variants of them / Analizar algoritmos de computación cuántica conocidos y posibles variaciones de éstos.
- RAS17: Evaluates the security of quantum cryptographic protocols / Valorar la seguridad de protocolos criptográficos cuánticos.

Competencies:

- RAC21: Assess and suggest cryptographic techniques to ensure the security and privacy in a computer system / Valorar y sugerir técnicas criptográficas para garantizar la seguridad y privacidad en sistemas informáticos.

LEARNING ACTIVITIES

	Total hours
Master classes	43
Problem solving	0
Laboratory Sessions	0
Oral presentations	4
Taking exams	1
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

The assessment method will consist of two parts:

- In-class exams: 30%.
- Assignments to be developed during the course and a term project that has to be presented: 70% .

BIBLIOGRAPHY

Basic bibliography:

M. Nielsen, I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

Other relevant bibliography:

J. Preskill, Quantum computation: lecture notes. Available at:
http://www.theory.caltech.edu/%7Epreskill/ph219/index.html#lecture

R. de Wolf, Quantum Computing: lecture notes. Available at: https://homepages.cwi.nl/~rdewolf/qcnotes.pdf

M.M. Wilde, From Classical to Quantum Shannon Theory. Available at: https://arxiv.org/pdf/1106.1445.pdf

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Curso 2025-2026

SUBJECT DESCRIPTION

Master in:	Formal Methods in Computer Science
Course:	610612 - Cryptographic protocols and applications (PCA)
Course in Spanish:	Protocolos criptográficos y sus aplicaciones
Year and term:	1st (Q1)
ECTS:	6
Type:	Elective
Teaching language:	English
Module:	Elective
Subject:	Rigorous systems design and construction
Departament:	Sistemas Informáticos y Computación
Coordinator of the course:	Albert Rubio Gimeno (at UCM)

REQUIRED CONTENTS

- Perfect security and computational security
- Pseudorandomness
- Symmetric-key cryptography
- Public key cryptography
- Advanced protocols
- Aplicaciones en casos reales.

DETAILED PROGRAM

1. Perfect security and computational security
   - 1.1. Notion of encryption, one-time pad
   - 1.2. Notions of security
   - 1.3. One-way functions and one way trapdoor functions, mathematical examples
   - 1.4. IND-CPA security
2. Pseudorandomness and symmetric-key cryptography:
   - 2.1. Hardcore predicates for one-way functions
   - 2.2. Pseudorandom generators and pseudorandom functions
   - 2.3. Message authentication codes
   - 2.4. Symmetric key encryption
   - 2.5. Modes of operation
3. Public key cryptography:
   - 3.1. Key Exchange
   - 3.2. Public key encryption
   - 3.3. El Gamal encryption
   - 3.4. RSA encryption
   - 3.5. Digital signatures
4. 4. Advanced protocols
   - 4.1. Zero knowledge proofs
   - 4.2. Secret sharing
   - 4.3. Secure multiparty computation
5. Applications in real world scenarios
   - 5.1. Blockchain technologies

DETAILED PROGRAM IN SPANISH

1. Seguridad perfecta y seguridad computacional:
   - 1.1. Noción de encriptación, libreta de un solo uso
   - 1.2. Nociones de seguridad
   - 1.3. Funciones unidireccionales y funciones unidireccionales trampa, ejemplos matemáticos
   - 1.4. Seguridad IND-CPA
2. Pseudoaleatoriedad y criptografía de clave simétrica:
   - 2.1. Predicados de núcleo duro para funciones unidireccionales
   - 2.2. Generadores pseudoaleatorios y funciones pseudoaleatorias
   - 2.3. Códigos de autenticación de mensajes
   - 2.4. Encriptación de clave simétrica
   - 2.5. Modos de operación
3. Criptografía de clave pública:
   - 3.1. Intercambio de claves
   - 3.2. Encriptación de clave pública
   - 3.3. Cifrado ElGamal
   - 3.4. Cifrado RSA
   - 3.5. Firma digital
4. Protocolos de seguridad avanzados:
   - 4.1. Pruebas de conocimiento nulo
   - 4.2. Secreto compartido
   - 4.3. Computación multipartita segura
5. Aplicaciones en situaciones del mundo real:
   - 5.1. Tecnologías blockchain

TRAINING RESULTS

Skills:

- RAS13: Argumentar mediante métodos matemáticos la idoneidad de una técnica criptográfica en un contexto de seguridad concreta / Argue through mathematical methods the suitability of a cryptographic technique in a specific security context.

Competencies:

- RAC21: Valorar y sugerir técnicas criptográficas para garantizar la seguridad y privacidad en sistemas informáticos / Assess and suggest cryptographic techniques to ensure the security and privacy in a computer system.

LEARNING ACTIVITIES

	Total hours
Master classes	33
Problem solving	7
Laboratory Sessions	4

Oral presentations	2
Taking exams	2
Total	**48 hours**
Independent study	102 hours
Total	**150 horas**

DETAILED EVALUATION

The assessment method will consist of two parts:
- Assignments to be developed during the course period at home [60%].
- In-class final exam [40%].

The students have the possibility to skip the assignments and take only an in-class final exam [100%].

Referred (re-sit) assessment: in-class exam [100%]

BIBLIOGRAPHY

Basic:
-Goldwasser, Bellare: Lecture Notes on Cryptography, https://cseweb.ucsd.edu/~mihir/papers/gb.pdf
-Boneh, Shoup: A Graduate Course in Applied Cryptography, http://toc.cryptobook.us/
Additional reading:
-Katz, Lindell: Introduction to Modern Cryptography, Second Edition, CRC Press
-Barak: An Intensive Introduction to Cryptography, https://intensecrypto.org/public/index.html
-Rosulek: The Joy of Cryptography, https://web.engr.oregonstate.edu/~rosulekm/crypto/