ANX-PR/CL/001-01

LEARNING GUIDE

SUBJECT

103000800 - Theory Of Programming Languages

DEGREE PROGRAMME

10AS - Máster Interuniversitario En Métodos Formales En Ingeniería Informática

ACADEMIC YEAR & SEMESTER

2025/26 - Semester 1

Index

Learning guide

1. Description

1.1. Subject details

Name of the subject	103000800 - Theory Of Programming Languages
No of credits	6 ECTS
Type	Compulsory
Academic year ot the programme	First year
Semester of tuition	Semester 1
Tuition period	September-January
Tuition languages	English
Degree programme	10AS - Máster Interuniversitario en Métodos Formales en Ingeniería Informática
Centre	10 - E.T.S. De Ingenieros Informáticos
Academic year	2025-26

2. Faculty

2.1. Faculty members with subject teaching role

Name and surname	Office/Room	Email	Tutoring hours *
Julio Mariño Carballo (Subject coordinator)	D2308	julio.marino@upm.es	Tu - 15:00 - 17:00 W - 17:00 - 19:00 F - 15:00 - 17:00 Supervision may take place in the "Facultad de Informática (UCM)" or online.

* The tutoring schedule is indicative and subject to possible changes. Please check tutoring times with the faculty

member in charge.

2.3. External faculty

Name and surname	Email	Institution
Narciso Martí Oliet	narciso@ucm.es	Facultad de Informática UCM
Ignacio Fábregas Alfaro	fabregas@fdi.ucm.es	Facultad de Informática UCM

3. Prior knowledge recommended to take the subject

3.1. Recommended (passed) subjects

The subject - recommended (passed), are not defined.

3.2. Other recommended learning outcomes

- Proof by induction. Instrumental knowledge of the induction principle and its associated proof techniques is required. Supplementary materials will be provided to students not proving that knowledge.

- Mathematical logic. Students not able to demonstrate basic knowledge will be required to acquire it with supplementary materials.

4. Skills and learning outcomes *

4.1. Skills to be learned

RAC18 - Definir y analizar la semántica formal de un lenguaje de programación / Define and analyse the formal semantics of a programming language. TIPO: Competencias.

RAC19 - Diseñar análisis estáticos de programas informáticos basados en interpretación abstracta, sistemas de tipos y resolución de restricciones / Design static analyses of computer programs based on abstract interpretation, type systems and constraint resolution. TIPO: Competencias.

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. TIPO: Competencias.

RAK1 - Comparar y valorar los sistemas de tipos en diferentes lenguajes de programación / Compare and evaluate

type systems in different programming languages. TIPO: Conocimientos o contenidos.

RAK2 - Comprender y analizar distintas formalizaciones de la semántica de un lenguaje de programación / Understand and analyse different formalizations of the semantics of a programming language. TIPO: Conocimientos o contenidos.

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. TIPO: Conocimientos o contenidos.

RAK4 - Comparar distintos modelos teóricos en el estudio de sistemas informáticos / Compare different theoretical models in the study of computer systems. TIPO: Conocimientos o contenidos.

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

RAK6 - Modelar problemas de análisis estático como problemas de resolución de restricciones matemáticas / Model static análisis problems as mathematical constraint solving problems. TIPO: Conocimientos o contenidos.

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

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

4.2. Learning outcomes

RA1 - All RA are defined in the previous section

* The Learning Guides should reflect the Skills and Learning Outcomes in the same way as indicated in the Degree Verification Memory. For this reason, they have not been translated into English and appear in Spanish.

5. Brief description of the subject and syllabus

5.1. Brief description of the subject

The course provides the formal background needed to reason about software and programming languages in a precise and mathematically sound way.
Fundamental concepts underlying the design, definition and execution mechanisms of programming languages are covered, including. rewriting, recursion, syntax, various forms of semantics and type systems.

Alongside the theoretical contents, the course may include small programming assignments to gain a more instrumental level of the ideas mentioned above.

5.2. Syllabus

1. Introduction and review of background
   1.1. Presentation of the course: aim and goals
   1.2. Review of background: formal logic, mathematical induction, abstract syntax.
2. Lambda calculi
   2.1. The untyped lambda calculus
   2.2. The simply typed lambda calculus
3. Type systems
   3.1. Natural deduction and the Curry-Howard isomorphism
   3.2. Polymorphism: lambda2
   3.3. Polymorphism: Hindley-Milner
4. Reduction and rewriting
   4.1. Abstract reduction systems
   4.2. Term rewriting systems
5. Semantics
   5.1. Operational semantics
   5.2. Domains and denotational semantics

6. Schedule

6.1. Subject schedule*

Week	Type 1 activities	Type 2 activities	Distant / On-line	Assessment activities
1	**Theory** Duration: 02:00 **Theory** Duration: 01:30			
2	**Theory** Duration: 02:00 **Theory** Duration: 01:30			
3	**Theory** Duration: 02:00 **Theory** Duration: 01:30			**Exercises** Progressive assessment Not Presential Duration: 04:00
4	**Theory** Duration: 02:00 **Exercise review** Duration: 01:30			
5	**Theory** Duration: 02:00 **Theory** Duration: 01:30			
6	**Theory** Duration: 02:00 **Theory** Duration: 01:30			**Exercises** Progressive assessment Not Presential Duration: 03:00

7	**Theory** Duration: 02:00 **Exercise review** Duration: 01:30			
8	**Theory** Duration: 02:00 **Theory** Duration: 01:30			
9	**Theory** Duration: 02:00 **Theory** Duration: 01:30			
10	**Theory** Duration: 02:00 **Theory** Duration: 01:30			
11	**Theory** Duration: 02:00 **Theory** Duration: 01:30			**Exercises** Progressive assessment Not Presential Duration: 04:00
12	**Theory** Duration: 02:00 **Exercise review** Duration: 01:30			
13	**Theory** Duration: 02:00 **Theory** Duration: 01:30			
14	**Theory** Duration: 02:00 **Theory** Duration: 01:30			**Exercises** Progressive assessment Not Presential Duration: 04:00

15	**Theory** Duration: 02:00 **Exercise review** Duration: 01:30			
16				
17	**Final exam** Duration: 03:00			**Final exam** Progressive assessment and Global Examination Presential Duration: 03:00

Depending on the programme study plan, total values will be calculated according to the ECTS credit unit as 26/27 hours of student face-to-face contact and independent study time.

7. Activities and assessment criteria

7.1. Assessment activities

7.1.1. Assessment

Week	Description	Modality	Type	Duration	Weight	Minimum grade	Evaluated skills
3	Exercises		No Presential	04:00	12.5%	3 / 10	RAK4 RAK5
6	Exercises		No Presential	03:00	12.5%	3 / 10	RAK2 RAK3 RAK4 RAK5 RAS8 RAC18 RAC20
11	Exercises		No Presential	04:00	12.5%	3 / 10	RAK2 RAK4 RAK5 RAS8 RAC18
14	Exercises		No Presential	04:00	12.5%	3 / 10	RAK1 RAK6 RAS7 RAS8
17	Final exam		Face-to-face	03:00	50%	4 / 10	RAK1 RAK2 RAK3 RAK4 RAK5 RAK6 RAS7 RAS8 RAC18 RAC19 RAC20

7.1.2. Global examination

Week	Description	Modality	Type	Duration	Weight	Minimum grade	Evaluated skills

17	Final exam		Face-to-face	03:00	50%	4 / 10	RAK1 RAK2 RAK3 RAK4 RAK5 RAK6 RAS7 RAS8 RAC18 RAC19 RAC20

7.1.3. Referred (re-sit) examination

Description	Modality	Type	Duration	Weight	Minimum grade	Evaluated skills
Final extraordinary exam		Face-to-face	03:00	100%	5 / 10	

7.2. Assessment criteria

Assessment is based on a combination of individual homework and a final exam (50/50).

8. Teaching resources

8.1. Teaching resources for the subject

Name	Type	Notes
Moodle site for the course	Web resource	Most of the course materials will be available at the "Virtual Campus" of the UCM.
Foundations for Programming Languages. John C. Mitchell. The MIT Press	Bibliography	
Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics. Jan van Leeuwen (ed.) The MIT Press	Bibliography	
Term Rewriting Systems. Terese. Cambridge Tracts in Theoretical Computer Science.	Bibliography	
The Formal Semantics of Programming Languages: An Introduction. Glynn Winskel. The MIT Press	Bibliography	
Basic Simple Type Theory. Roger Hindley. Cambridge Tracts in Theoretical Computer Science.	Bibliography	
Proofs and Types. Girard, Lafont and Taylor. Cambridge Tracts in Theoretical Computer Science.	Bibliography	
Types and Programming Languages. Benjamin Pierce. The MIT Press	Bibliography	
Baader, F. & Nipkow, T. Term rewriting and all that. Cambridge University Press.	Bibliography	

Clavel, M. et al. All about Maude - A high performance logical framework. Lecture Notes in Computer Science 4350. Springer Verlag.	Bibliography	
Nielson, H. R.; Nielson, F. Semantics with Applications : an Appetizer; Undergraduate Topics in Computer Science; Springer Verlag.	Bibliography	
Ford, R. Leino, K. Rustan M. Dafny Reference Manual.	Bibliography	Complementaria
The Lambda Calculus: its Syntax and Semantics. Henk Barendregt. Studies in Logic and the Foundations of Mathematics, vol. 103. North Holland.	Bibliography	Complementaria

9. Other information

9.1. Other information about the subject

The course is taught at UCM's Facultad de Informática (School of Computer Science). UCM regulations apply.