MODELLO D (inglese)	
General Information	Formal Methods in Computer Science
Academic subject	Theoretical Computer Science
Degree course	Computer Science
Curriculum	6
ECTS credits	
Compulsory attendance	No
Language	English

Subject teacher	Name Surname	Mail address	SSD
	Giovanni Pani	**giovanni.pani@uniba.it**	Inf01

ECTS credits details			
Basic teaching activities	Lectures	videos	laboratory

Class schedule	
Period	First term
Year	First
Type of class	Lecture, laboratory, videos.

Time management	
Hours	150
Hours of lectures	32
Tutorials and lab	30

Academic calendar	
Class begins	25th September 2018
Class ends	12th January 2019

Syllabus	
Prerequisites/requirements	Programming languages Computability and Complexity
Expected learning outcomes (according to Dublin Descriptors) (it is recommended that they are congruent with the learning outcomes contained in A4a, A4b, A4c tables of the SUA-CdS)	*Syntax and semantics of programming languages.* *Applying knowledge to define programming languages and to define interpreter using functional programming.* *Choice a programming languages.* *Design and develop software, also suggesting to evaluating alternative solutions and define most appropriate programming languages.* *Capacities to continue learning.*
Contents	Formal Semantics of Programming languages. Functional Programming.
Course program	Theory. Basic set Theory. Logical notations. Sets. Relations and functions. The axiom of foundation. Lambda notations. Composing relations and functions. Direct and inverse image of a relation. Equivalence relations.

	Introduction to operational semantics. IMP a simple imperative languages. The evaluation of arithmetic and Boolean expressions; the execution of commands. Some principles of induction. Mathematical induction, Structural induction, well founded induction. Inductive definitions. Well founded induction. Induction an derivations. Definition by induction. Inductive definitions. Rule induction, Special rule induction. Proof rules for operational semantics. Rule induction for Arithmetical expressions. Rule induction for Boolean expressions. Rukle induction for commands. Operators and their least fixed points. The denotational semantics of IMP. Denotational semantics. Equivalence of the semantics. Complete partial orders and continuous functions. The Knaster Tarski theorem. Introduction to domains theory. Basic definitions. Streams, an example. Costruction on cpo's. Discrete cpo's. Finite products, Function space. Lifting. Sums. A metalanguage. Recursion equations. The language REC. Operations and denotational semantics for call by value. Equivalence of the two semantics. Operations and denotational semantics for call by name. Equivalence of the two semantics. Languages with higher types. An Eager language. Eager operational and denotational semantics. Agreement of the two semantics (no proof). A lazy language. Lazy operational and denotational semantics. Agreement of the two semantics (no proof). Fixed point operators. Laboratory. Haskell. Functions. Functional programming. Features of Haskell. The hugs system. The standard prelude. Functional application. Haskell scripts. Types and classes. Basic concepts. Basic types. List types. Tuple types. Function types. Curried Functions.Polimorphic types. Overloaded types. Basic classes. Functions. Defining functions. Conditional expressions. Guarded equations. Pattern matching. Lambda expressions. Sections. List comprehensions. Generators. Guards. The zip function. String comprehensions. Recursive functions. Recursions on lists. Multiple arguments. Multiple recursions. Mutual recursions. Higher order functions. Processing lists. The foldr function. The foldl function. The composition operator.

	Functional parser and monads. Parser. The parser type. Basic parsers. Sequencing. Choice. Derived primitives. Hamdling spaces. Arithmetical expressions. Interactive programs. Interaction. The input output type. Basic actions. Sequencing. Derived primitives. Declarating types and classes. Type declarations. Data declarations. Recursive types. Abstract machine. Class and instance declarations. List comprehensions. Recursive functions. Higher order functions. Functional parser. Interpreter.
Bibliography	G. Winskel , The formal semantics of programming languages, Mit press. G. Hutton, Programming in Haskell, (II edition) Cambridge University Press.
Notes	
Teaching methods	Lectures, videos, laboratory
Assessment methods (indicate at least the type written, oral, other)	Laboratory. Theoretical part exam either written or oral
Evaluation criteria (Explain for each expected learning outcome what a student has to know, or is able to do, and how many levels of achievement there are.	Marks. Laboratory minimum 6 max 10 plus theoretical part minimum 12 max 20.
Further information	