SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeş Bolyai University**
1.2 Faculty	**Faculty of Mathematics and Computer Science**
1.3 Department	**Department of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Artificial Intelligence**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Automata Theory and Compilers**				
2.2 Course coordinator			**Prof.PhD. Simona Motogna**				
2.3 Seminar coordinator			**Prof.PhD. Simona Motogna**				
2.4. Year of study	**3**	2.5 Semeste	**5**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**Compulsory**
2.8 Code of the discipline		MLE5206					

3. Total estimated time (hours/semester of didactic activities)

3.1 Hours per week	6	Of which: 3.2 course	2	3.3 seminar/laboratory	2sem + 2lab
3.4 Total hours in the curriculum	84	Of which: 3.5 course	28	3.6 seminar/laboratory	56

Time allotment:	hours
Learning using manual, course support, bibliography, course notes	10
Additional documentation (in libraries, on electronic platforms, field documentation)	5
Preparation for seminars/labs, homework, papers, portfolios and essays	10
Tutorship	6
Evaluations	10
Other activities:	-

3.7 Total individual study hours	41
3.8 Total hours per semester	125
3.9 Number of ECTS credits	5

4. Prerequisites (if necessary)

4.1. curriculum	• Fundamentals of Programming, Data Structures and Algorithms

4.2. competencies	• Average programming skills in a high level programming language

5. Conditions (if necessary)

5.1. for the course	• Course roo with projector
5.2. for the seminar /lab activities	• Laboratory with computers

6. Specific competencies acquired

Professional competencies	• C4.1 Definition of concepts and basic principles of computer science, and their mathematical models and theories • C4.2 Interpretation of mathematical and computer science models • C4.3 Identify adequate models and methods to solve real life problems • C4.5 Adoption of formal models in specific applications from different domains
Transversal competencies	CT1 Apply rules to: organized and efficient work, responsabilities of didactical and scientifical activities and creative capitalization of own potential, while respecting principles and rules for professional ethics CT3 Use efficient methods and techniques for learning, knowledge gaining, and research and develop capabilities for capitalization of knowledge, accomodation to society requirements and communication in English

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• Knowledge, understanding an duse of theoretical concepts used in compiler design • Improved programming skills
7.2 Specific objective of the discipline	• Acquire knowledge about back-end of a compiler • Improved programming skills: understang the underleying functioning of a compiler, program debugging, better compiling error reporting • Understading of formal langauges concepts and development of skills to model problems using formal languages; ability to apply compiler specific techniques to diverse real life problems

8. Content

8.1 Course	Teaching methods	Remarks
1. General Structure of a compiler. Introduction	Exposure: description, explanation, examples	
2. Scanning (Lexical Analysis). Formal Languages	Exposure: description, explanation, examples	
3. Grammars. Chomsky classification. Finite Automata	Exposure: description, explanation, examples	
4. Regular languages. Sanner generators	Exposure: description, explanation, examples	
5. Closure properties for regular languages	Exposure: description, explanation, examples	
6. Context-free grammars	Exposure, explanation,	

	examples, discussion of case studies	
7. Parser generators. Push Down Automata	Exposure: explanation, examples, discussion of case studies	
8. Attribute grammars	Exposure: description, explanation, examples	
9. & 10 Parsing (Syntactical Analysis)	Exposure: description, explanation, examples	
11. & 12 Syntax Tree. Intermediary Code	Exposure: description, examples	
13. & 14 Summarization of theoretical and practical aspects. Application in compiler design	Exposure: description, explanation, examples, demo	

Bibliography

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. D. GRIES - Compiler construction for digital computers,, John Wiley, New York, 1971.
4. MOTOGNA, S. – Metode de proiectare a compilatoarelor, Ed. Albastra, 2006
5. SIPSER, M., Introduction to the theory of computation, PWS Pulb. Co., 1997
6. CSÖRNYEI ZOLTÁN, Bevezetés a fordítóprogramok elméletébe, I, II., ELTE, Budapest, 1996
7. L.D. SERBANATI - Limbaje de programare si compilatoare, Ed. Academiei RSR, 1987.

8.2 Seminar	Teaching methods	Remarks
1. Specification of a programming language; BNF notation	Explanation, dialogue, case studies	
2. Finite automata: language generated by a FA; FA corresponding to a language	Dialogue, debate, case studies, examples, proof	
3. Finite automata: data structures for finite automata	Dialogue, debate, case studies, examples	
4. Properties of regular languages. Proofs and applications	Dialogue, debate, case studies, examples	
5. Grammars; language generated by a grammar; grammar corresponding to a language	Dialogue, debate, case studies, examples, proof	
6. LR(0), SRL parsing	Dialogue, debate, case studies, examples	
7. LR(1) parsing	Dialogue, debate, case studies, examples	
8. LALR parsing	Dialogue, debate, case studies, examples	
9. Context free grammars. Push Down Automata	Dialogue, debate, case studies, examples	
10. LL(1) parsing	Dialogue, debate, case studies, examples	
11. Descendent recursive parser	Dialogue, debate, case studies, examples	
12. & 13 Properties of context free grammars. Proofs and applications	Dialogue, debate, case studies, examples	
14. Summarization exercices (grammar of types 1,2, and 3)	Dialogue, debate, case studies, examples	

8.3 Laboratory	Teaching methods	Remarks
1. Task 1: Specify a mini-language and implement scanner 1.1: Mini language specification (BNF notation)	Explanation, dialogue, case studies	
2. Task 1: Specify a mini-language and implement scanner 1.2: implement main functions in scanning	Explanation, dialogue, case studies	
3. Task 1: Specify a mini-language and implement scanner 1.3: Symbol Table organization	Explanation, dialogue, case studies	
4. Task 1: Specify a mini-language and implement scanner 1.4: Main program, testing + delivery	Testing data discussion, evaluation	
5. Task 2: Finite Automata 2.1: Verify sequence acceptance DFA and NFA	Explanation, dialogue, case studies	
6. Task 2: Finite Automata 2.2: Adapt scanner to use FA to determine tokens	Testing data discussion, evaluation	
7. Task 3: Parser implementations 3.1: define data structures and architecture of application 3.2 implement main functions in parsing	Explanation, dialogue, case studies	One of: descendant recursive, LL(1), LR(0), SLR
8. Task 3: Parser implementations 3.3: main program and module integration	Testing data discussion, evaluation	Task 3 is developed in teams of 2 students
9. Task 3: Parser implementations 3.4: testing and error handling	Explanation, dialogue, case studies	
10. Task 3: Parser implementations 3.5: delivery	Explanation, dialogue, case studies	
11. Task 4: use tools for lexer generator: lex,	Explanation, dialogue, case studies	
12. Task 5: use tools for parser generator: yacc	Testing data discussion, evaluation	
13. Task 6: use tools for lexer and parser generator 6.1 Combine the 2 tools and re-run tasks 3 and 4	Testing data discussion, evaluation	
15. Task 6: use tools for lexer and parser generator: 6.2 Testing and delivery	Explanation, dialogue, case studies	

Bibliography

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. MOTOGNA, S. – Metode de proiectare a compilatoarelor, Ed. Albastra, 2006
4. G. MOLDOVAN, V. CIOBAN, M. LUPEA - Limbaje formale si automate. Culegere de probleme, Univ. Babes-Bolyai, Cluj-Napoca, 1996.
5. D. GRIES - Compiler construction for digital computers,, John Wiley, New York, 1971.
6. L.D. SERBANATI - Limbaje de programare si compilatoare, Ed. Academiei RSR, 1987.

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course respects the IEEE and ACM Curriculla Recommendations for Computer Science studies;
- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course is considered the software companies as important for average programming skills

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)

10.4 Course	- know the basic principle of the domain; - apply the course concepts - problem solving	Written exam	60%
10.5 Seminar and lab activities	- be able to apply algorithms, understand examples - problem solving	problems solved - homeworks delivered - continuous observations during semester	10%
	- be able to implement course concepts and algorithms - apply techniques for different classes of programming languages	-Practical examination during all semester -documentation - portofolio -continous observations	30%
10.6 Minimum performance standards			
➢ Attend 75% of seminar activities during semester AND attend 90% of lab activities during semester ➢ At least grade 5 (from a scale of 1 to 10) at both written exam and laboratory work.			

Date	Signature of course coordinator	Signature of seminar coordinator
25.04.2023	Prof.PhD. Simona MOTOGNA	Prof.PhD. Simona MOTOGNA

Date of approval ..... Signature of the head of department

Prof.dr. Laura Dioșan