SYLLABUS

Automata Theory and Compilers

University year 2025-2026

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**
1.7. Form of education	**Full time**

2. Information regarding the discipline

2.1. Name of the discipline	**Automata Theory and Compilers**					Discipline code	**MLE5206**
2.2. Course coordinator					**Prof.PhD. Simona Motogna**		
2.3. Seminar coordinator					**Prof.PhD. Simona Motogna**		
2.4. Year of study	3	2.5. Semester	5	2.6. Type of evaluation	E	2.7. Discipline regime	Mandatory

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/project	**2+2**
3.4. Total hours in the curriculum	84	of which: 3.5 course	28	3.6 seminar/laboratory/project	**56**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					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	Programming fundamentals, Data structures and algorithms
4.2. competencies	Medium programming skills in a high level programming language

5. Conditions (if necessary)

5.1. for the course	Room with projector
5.2. for the seminar /lab activities	Computers/laptops Licensed programming software (.NET, Java, Python or similar)

6.1. Specific competencies acquired [1]

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

Professional/essential competencies	• develop the prototype for the software • design the IT system • fix errors in the software
Transversal competencies	• work in teams • think analytically

6.2. Learning outcomes

Knowledge	- The graduate knows and understands the mathematical foundations needed to develop intelligent algorithms and is capable of using them for algorithm implementation. - The graduate has knowledge of programming, mathematics, engineering and technology and has the skills to use them in creating complex computer systems.
Skills	- The graduate is able to evaluate, both quantitatively and qualitatively, the performance of intelligent systems. - The graduate is able to identify complex issues and examine related issues in order to design several solutions and implement these solutions.
Responsibility and autonomy:	- The graduate has the ability to choose and use programming paradigms (procedural, object-oriented, functional) to create software applications appropriate to the specific field of the developed application. - The graduate has the necessary skills to apply various methods and tools for analysis and visualizing the results of the used Artificial Intelligence algorithms and techniques.

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

7.1 General objective of the discipline	• Knowledge, understanding and use 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: understand the underlying 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, demo	
2. Scanning (Lexical Analysis). Formal Languages	Exposure: description, explanation, examples, demo	
3. Grammars. Chomsky classification. Finite Automata	Exposure: description, explanation, examples, demo	
4. Regular languages. Scanner generators	Exposure: description, explanation, examples, demo	
5. Closure properties for regular languages	Exposure: description, explanation, examples, demo	
6. Context-free grammars	Exposure: description, explanation, examples, demo	
7. Parser generators. Push Down Automata	Exposure: description, explanation, examples, demo	
8. Attribute grammars	Exposure: description, explanation, examples, demo	
9 & 10 Parsing (Syntactical Analysis)	Exposure: description, explanation, examples, demo	
11 & 12 Intermediary code and object code generation	Exposure: description, explanation, examples, demo	
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	Dialogue, debate, case studies, examples	
2. Finite automata	Dialogue, debate, case studies, examples	
3. Regular and context free grammars	Dialogue, debate, case studies, examples	
4 & 5 Properties of regular languages	Dialogue, debate, case studies, examples	
6. LR(0)parsing	Dialogue, debate, case studies, examples	
7. SLR parsing	Dialogue, debate, case studies, examples	
8. LR(1) and LALR parsing	Dialogue, debate, case studies, examples	
9. Push Down Automata	Dialogue, debate, case studies, examples	
10. LL(1) parsing	Dialogue, debate, case studies, examples	
11. Attribute grammars	Dialogue, debate, case studies, examples	
12. Intermediary code	Dialogue, debate, case studies, examples	
13. Properties of cfg	Dialogue, debate, case studies, examples	
14. Summarization exercices	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	
14. 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

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade

10.4 Course	- know the basic principle of the domain; - apply the course concepts - problem solving	Written exam	60%
10.5 Seminar/laboratory	- 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 standard of performance			
➢ 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. • Understand the basic concepts of formal languages: grammar, FA, PDA, regular expressions; understand compiling principles, scanning and parsing			

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date:

12.04.2025

Signature of course coordinator

Prof.PhD. Simona Motogna

Signature of seminar coordinator

Prof.PhD. Simona Motogna

Date of approval:
...

Signature of the head of department

Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable*.”.