# SYLLABUS

**Name:** ***Automatic inference systems (AiRAu>SI4-AIS-19)***
**Name in Polish:**
**Name in English:** ***Automatic inference systems***

## Information on course:

**Course offered by department:** Faculty of Automatic Control, Electronics and Computer Science
**Course for department:** Silesian University of Technology

**Default type of course examination report:**

ZAL

**Language:**

English

**Course homepage:**

https://platforma2.polsl.pl/rau1/course/view.php?id=83

**Short description:**

The aim of the course is to present automatic inference techniques for solving constraint satisfaction and optimisation combinatorial problems, such as scheduling or planning problems. Examples of such a problems are solved using logic programming (Prolog language) and object-oriented programming (Java language supported by JaCoP library for Constraint Logic Programming). The theory presented in the lecture is practised during laboratory exercises using dedicated IT tools. Pre-requisite qualifications include knowledge of procedural programming and object-oriented languages (courses: Programowanie obliczeń komputerowych, Programowanie obiektowe) as well as basic concepts of mathematical logic (Algebra).

**Description:**

ECTS: 3
Total workload: 80 hours (50 contact hours, 30 students' own work hours)
Forms of contact hours:
Lecture 30h
Laboratory 15h
Other (e.g. test and reports revision and discussion) 5h
Students' own work: preparation for classes, writing reports, preparation for tests

Lecture:
Constraint Logic Programming is a methodology having backgrounds in Operational Research and belongs to wide concept of Artificial Intelligence Methods. Its main concept is based on declarative programming – the programmer needs only to describe what computation should be performed and not how to compute it (this part is already implemented in the solver). Lecture is conducted with a multimedia presentation (available for the course students), supported with example code testing. The course presents the following concepts:
1. Declarative/logic programming concepts based on Prolog language (structure of program, types of data, logical inference: unification and backtracking, standard predicates, recursion, dynamic databases, solving combinatorial problems by brute-force search (generate and test method) and using backtracking, the problem of combinatorial explosion, classification of problems solved in constraint logic programming),
2. Advances in languages dedicated for solving constraint logic programming problems on the basis of ECLiPSe solver vs. “pure” Prolog,
3. Solving constraint logic programming problems in Java language with dedicated JaCoP (Java Constraint Programming) library (finite domain variables, primitive and global constraints, constraint propagation, modelling and solving specific classes of problems – combinatorial problems, scheduling/planning problems, path optimisation problems, search strategies, search plug-ins).
Details of concepts mentioned above are presented while solving examples of simple academic and more complex, real-world problems. Advantages of using well known and well documented language allows the students to focus on getting to know the idea behind constraint programming.

Laboratory:
The laboratory is divided into two blocks with two different IT tools: Prolog (1, 2, 3) and Java+JaCoP (4, 5, 6):
1. Structure of program and logical inference in logic programming. Elementary logical operations in Prolog. Entering query and searching for all answears, using standard predicates to enforce or prevent backtracking.
2. Recursion – types of recursions with examples. Lists – structure, access to elements, creating and sorting lists.
3. Combinatorial problems – brute-force method vs. standard backtracking method. Modelling and searching for all the feasible solutions.
4. Combinatorial problems in Java+JaCoP – modelling and search parametrisation.
5. Task schedulling problems in Java+JaCoP – diffn and cumulative contraints.
6. Optimisation in Java+JaCoP.

**Bibliography:**

Barták, R.: Guide to Prolog Programming, http://kti.mff.cuni.cz/~bartak/prolog/
Learn Prolog Now! http://www.learnprolognow.org/
Francesca Rossi, Peter Van Beek, Toby Walsh “Handbook of constraint programming”
Peter van Hentenryck “Constraint-based Local Search”
http://www.jacop.eu (JaCoP API and documentation)

**Learning outcomes:**

Course-specific learning outcomes - at the completion of the course, student:
- knows the tasks of constraint programming, the methods and tools used to achieve the goal, and their possible applications (entrance test, final test) K1A_W16
- knows the structure of the program, its components, basic standard predicates, understands the mechanisms of inference in Prolog, in particular backtracking, recursion, and operations on lists (entrance test, final test, laboratory report) K1A_W16
- knows and understands meaning of: modelling, finite domain variable, types of constraints, constraints propagation, search methods and their functionalities (entrance test, final test, laboratory report) K1A_W16
- can predict the result of the program, and write his own procedures in accordance with the principles of programming (laboratory report) K1A_U22
- is able to apply the knowledge to simple problems of combinatorial programming, in particular for problems of the another kind than in the examples given during the lectures (laboratory report) K1A_U22

**Assessment methods and assessment criteria:**

Course consists of two components: lecture and laboratory. According to SUT regulation, lecture attendance is optional (however highly recommended), whereas laboratory exercises are obligatory.

Course is assessed based on the final laboratory grade. Additionally, during the last lecture in the semester there is an optional test for willing students who have obtained a positive final laboratory assessment (3.0 or more) and want to improve it. Passing the test (i.e. obtaining more than 50% of the points) allows for increasing the final course assessment by 0.5 (>50% of the points) or by 1 (>75% of the points) comparing to the final laboratory assessment (with the obvious upper limit of 5.0 according to the SUT grading scale). Students who do not take the test or fail it will receive a final course assessment equal to the final laboratory assessment.

Laboratory assessment:
Each of 6 obligatory laboratory exercises is graded from 2 to 5 (2 = unsatisfactory, 3 = satisfactory, 4 = good, 5 = very good) or 0 for unexcused absence. The resulting single exercise grade is affected by:
- the theoretical preparation for the exercise that can be checked with the short entrance test (written individually),
- the work performed during the laboratory (in 2-person sections),
- an obligatory report (made according to the teacher's requirements) uploaded on the Distance Education Platform at the end of each laboratory exercise (one per section),
- some exercises require additional supplement to the report, completed at home within a maximum of 1 week and uploaded on the Distance Education Platform.

The impact of each of the above components on the single exercise grade may vary for different exercises and is announced at the beginning of the exercise by the teacher.
Final laboratory grade is calculated as an average of grades obtained from all 6 exercises, rounded to the SUT grading scale, provided that additional conditions (related to the achievement of learning outcomes) are met: both averages calculated separately for two blocks of exercises (1-3 and 4-6) must be positive (i.e. 3.0 or more).

Making up missed laboratory exercises is possible only for students who have excuses for their absence on the date specified in the schedule or agreed with the teacher.

The syllabus is valid from academic year 2024/25 and its content cannot be changed during the semester.

## Course credits in various terms:

**<without a specific program>**

| Type of credits | Number | First term | Last term |
|---|---|---|---|
| European Credit Transfer System (ECTS) | 3 | 2020/2021-L | |