COURSE DESCRIPTION CARD - SYLLABUS

Course name
Constraint Programming [S2SI1E>POG]

Course

Field of study
Artificial Intelligence

Year/Semester
2/3

Area of study (specialization)
–

Profile of study
general academic

Level of study
second-cycle

Course offered in
English

Form of study
full-time

Requirements
elective

Number of hours

Lecture
15

Laboratory classes
15

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

2,00

Coordinators

dr inż. Adam Meissner
adam.meissner@put.poznan.pl

Lecturers

Prerequisites

A student has a basic knowledge of computational logic, declarative programming, imperative programming and combinatorial optimization. A student is able to communicate in English and to read descriptions and manuals of software tools, applications and similar documents. A student understands a responsibility associated to her/his own work. A student is able to adhere to team work rules and to take responsibility for cooperative tasks.

Course objective

The main goal of the course is to provide a student with fundamental concepts and methods of Constraint Programming (CP) and also to develop student's skills in applying this approach to model and to solve both theoretical and practical problems.

Course-related learning outcomes

Knowledge [K2st_W3] A student has advanced detailed knowledge regarding selected issues in artificial intelligence and related fields.
[K2st_W4] A student has knowledge about development trends and the most important cutting edge achievements in computer science, artificial intelligence and other selected and related scientific disciplines.

Skills [K2st_U1] A student is able to obtain information from literature, databases and other sources (both in Polish and English), integrate them, interpret and critically evaluate them, draw conclusions and formulate and fully justify opinions.
[K2st_U3] A student is able to plan and carry out experiments, including computer measurements and simulations, interpret the obtained results and draw conclusions and formulate and verify hypotheses related to complex engineering problems and simple research problems.
[K2st_U5] A student can - when formulating and solving engineering tasks - integrate knowledge from different areas of computer science and artificial intelligence (and if necessary also knowledge from other scientific disciplines) and apply a systemic approach, also taking into account non-technical aspects.
[K2st_U16] A student can determine the directions of further learning and implement the process of self-education, including other people.
Social competences [K2st_K2] A student understands the importance of using the latest knowledge in the field of computer science and artificial intelligence in solving research and practical problems.
[K2st_K4] A student is aware of the need to develop professional achievements and comply with the rules of professional ethics.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:
a) lectures - based on answers to questions concerning issues presented during lectures;
b) laboratories - based on student's activity in solving programming tasks.

Summative assessment:
a) lectures - based on a written test comprising ca. 5 questions covering both theoretical issues and simple tasks; the condition for passing the test is to obtain at least 50% of the total possible points.
b) laboratories - based on a test consisting in writing (ca. 3) programs.

Programme content

The program comprises the following issues: a general constraint satisfaction problem (CSP), basic methods for solving CSPs, classes of CSPs, types of constraints, constraint programming (CP) languages and environments, applications of CP.

Course topics

The lecture program comprises the following issues.
1) A definition and characteristic of CSP, basic rules of formulating CSPs
2) Basics of the MiniZinc language for formulating (programming) CSPs
3) General methods for solving CSPs - propagation and distribution of constraints, a search tree and its exploration by exact and heuristic algorithms
4) Improvements in solving CSPs - elimination of symmetries and introduction of redundant constraints
5) Constraint optimization problems and solving them by a two-dimensional distribution strategy
6) Selected types of constraints - global and reified constraints
7) Exemplary CSPs - logic puzzles, graph colouring, combinational circuit design, etc.
8) Application of CP to hard real-size problems
9) Advantages and drawbacks of CP - golden rules and tips for programmers
10) An overview of CP languages and development environments.

The introductory part of the lab exercises covers the basics of the MiniZinc IDE programming environment (its current version). During subsequent exercises, the environment will be used to formulate and solve CSP problems in the following areas.
1) Cryptarithms and alphametics
2) Chess problems
3) Magic squares and their variants (eq. latin squares)
4) Logic puzzles
5) Modelling of simple combinational circuits
6) Graph colouring

Teaching methods

Lectures: multimedia presentations supported by examples presented on the writing board.
Laboratories: programming task solving (individually or in small teams), discussion of student solutions.

Bibliography

Basic 1. Rossi F., Beek van F., Walsh T. (eds), Handbook of Constraint Programming. 1st Edition, Elsevier, 2006.
Additional 1. Apt K.R., Principles of Constraint Programming, Cambridge University Press, 2003.
2. Van Hentenryck P., Michel L., Constraint-Based Local Search, The MIT Press, 2005.
3. Stuckey P.J., Mariott K., Tack G., MiniZinc Handbook, (https://www.minizinc.org/).

Breakdown of average student's workload

	Hours	ECTS
Total workload	50	2,00
Classes requiring direct contact with the teacher	30	1,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	20	1,00