Government of Russian Federation

Federal State Autonomous Educational Institution of High Professional Education

"National Research University Higher School of Economics"

Syllabus for the course "Fundamentals of Knowledge Representation"

for postgraduate program in 02.06.01 Computer and Information science / 05.13.11 "Mathematical Theory and Software for Computing Machinery, Systems, and Networks", 05.13.17 "Theoretical Foundations of Computer Science"

Authors:
Max I. Kanovich, Dr.Sc., Professor, mkanovich@hse.ru

Approved by the Academic Council of the School for Postgraduate Studies in Computer Science by October 19, 2015

Moscow – 2015

1. Scope of Use

This program establishes the minimal requirements to postgraduate students' knowledge and skills for 02.06.01 Computer and Information science / 05.13.11 "Mathematical Theory and Software for Computing Machinery, Systems, and Networks", 05.13.17 "Theoretical Foundations of Computer Science", and determines the content of the course and educational techniques used in teaching the course.

The present syllabus is aimed at faculty teaching the course and postgraduate students studying 02.06.01 Computer and Information science / 05.13.11 "Mathematical Theory and Software for Computing Machinery, Systems, and Networks", 05.13.17 "Theoretical Foundations of Computer Science".

This syllabus meets the standards required by:

- Educational standards of National Research University Higher School of Economics;
- Postgraduate educational program for 02.06.01 Computer and Information science.
- University curriculum of the postgraduate program for 02.06.01 Computer and Information science / 05.13.11 "Mathematical Theory and Software for Computing Machinery, Systems, and Networks", 05.13.17 "Theoretical Foundations of Computer Science", approved in 2015.

2. The aim of the course. Learning Objectives.

Nowadays the knowledge representation systems are dealing with the representation the information in a form that a computer can utilize to provide an efficient human-computer interaction. The aim of this advanced course is to introduce the students to the most successful logic based *concepts, tools and techniques* used today in CS and IT, which are behind a major breakthrough in the theoretical and practical applications in knowledge representation systems. In this course we address the following issues which are proven to be of great theoretical and practical potential in CS and IT.

(a) Knowledge representation and reasoning in a static context.

Here predicate logic serves as a universal symbolic language to express the basic constructs such as *variables*, *properties*, *relations*, *quantifiers*, *inference rules of a universal nature*, etc.

(b) *Knowledge representation and reasoning in a dynamic context.*

For transition systems in AI and CS, here we use a language provided by Hoare triples to specify the dynamic correlation between their inputs and outputs, their preconditions and postconditions.

(c) *Knowledge representation: Tractable solutions to generally intractable problems.*

Notwithstanding that pure predicate logic is generally undecidable, we will discuss efficient algorithms to sort out certain problems of theoretical and practical interest.

This course is *new*. The course has been designed in accordance with the most recent trends in knowledge representation.

The challenge has been twofold:

(1) to select the material and design the course so that to make it meet the actual needs of the CS and AI applications, and

(2) to do that so that to allow the PhD students to learn and digest all necessary ideas to efficiently accommodate modern trends in knowledge representation.

This course is based on many years of the author's teaching experience in related subjects at the University of Pennsylvania and Queen Mary, University of London.

The course will be delivered in both languages, if necessary, in order

(a) to provide better understanding for the native-speaking people, and, at the same time,

(b) to guarantee a quicker adaptation to the international terminology, literature, the most innovative cutting edge techniques, etc., and

(c) to be prepared to understand / develop / use the most advanced automated tools in CS and IT.

3. Learning Outcomes

After studying the course, the PhD student should

- Learn syntax and semantics of predicate logic with ensuring its orientation to the actual needs of computer science and information technology;
- Understand how predicate logic detects difference between finite and infinite;
- Understand why predicate logic cannot detect difference between enumerable and non-enumerable;
- Be able to prove or disprove predicate formulas by means of the unfolded tree analysis;
- Be able to prove soundness and completeness for some finite systems of inference rules;
- Be able to simulate computations within predicate logic;
- Be familiar with decidable procedures for propositional logic;
- Be familiar with decidable procedures for pure monadic predicate logic;
- Be able to specify programs formally using Hoare triples;
- Be able to manually run symbolic execution rules for loop-free programs;
- Be able to use symbolic execution rules to prove loop-free programs semi-automatically;
- Understand the definition of loop invariants;
- Be able to write loop invariants for simple arithmetic programs;
- Be able to prove simple arithmetic programs using loop invariants;
- Understand symbolic execution for loops;
- Understand an algorithm for inferring loop invariants automatically;
- Be able to apply the algorithm for simple arithmetic programs;
- Understand SAT solvers;
- Understand the DPLL algorithm;
- Be able to apply Conflict-Driven Clause Learning as an efficient tool for SAT solving in practice.

After completing the study of the discipline the PhD student should have developed the following competencies:

Competence	**Code**	**Descriptors (indicators of achievement of the result)**	**Educative forms and methods aimed at generation and development of the competence**
The ability to critically analyze and evaluate research results including those obtained in multidisciplinary areas.	УК-1	The PhD student is able to carry out comparative testing of competing models or algorithms.	Examples covered during the lectures and tutorials.
The ability to generate original theoretical constructions, hypotheses, and research questions.	УК-2	The PhD student is able to suggest knowledge representation models for a particular real-life setup and reasoning algorithms for a particular knowledge representation language.	Tutorials.
The ability to carry out	ОПК-1	Students obtain necessary	Lectures and tutorials.

research in the field of professional activity using current research methods and information and communication technologies.		knowledge to understand and formulate the theoretical difficulty of problems they are solving.	
The ability to do research in transformation of information into data and knowledge, models of data and knowledge representation, methods for knowledge processing, machine learning and knowledge discovery methods, principles of building and operating software for automation of these processes.	ПК-4	Students learn current approaches to knowledge representation and are able to evaluate the complexity of the corresponding reasoning algorithms.	Lectures, tutorials.
The ability to develop and analyze models of information processes and structures.	ПК-5	The PhD student is able to select and adapt knowledge representation languages and reasoning algorithms for particular problems.	Lectures, tutorials.

4. Place of the Discipline in the Postgraduate Program Structure

This is an elective course for 05.13.11 "Mathematical Theory and Software for Computing Machinery, Systems, and Networks", 05.13.17 "Theoretical Foundations of Computer Science".

A recent knowledge of logic and math is needed.

5. Schedule

Topic	Total hours	Contact hours		Self-study
		Lectures	Tutorials	
Topic 1. Formal systems. Syntax and semantics.	14	2	2	10
Topic 2. Predicate logic as a specification language for AI and CS.	16	2	2	12
Topic 3: Models. Semantics of predicate logic. Finite and infinite models.	14	2	2	10
Topic 4: Automatized provers for predicate logic: Soundness and completeness.	14	2	2	10
Topic 5: Predicate logic: Undecidability.	14	2	2	10

Decidable fragments.				
Topic 6: Hoare triples as a language to specify the dynamic behaviour of programs/plans in AI and CS.	14	2	2	10
Topic 7: Hoare logic. Inference rules. Soundness and Completeness. Automatized Inference of Loop Invariants.	14	2	2	10
Topic 8: SAT Solvers. Symbolic model checking. Horn clauses and functional dependencies in relational databases. The DPLL algorithm and its improvements.	14	2	2	10
Total	114	16	16	82

6. Assessment

(i) Two mid-term tests worth 40% of the course marks.
(ii) Final exam worth 60% of the course marks.

Table of Grade Accordance

Ten-point Grading Scale	**Five-point Grading Scale**	
1 - very bad 2 – bad 3 – no pass	Unsatisfactory - 2	**FAIL**
4 – pass 5 – highly pass	Satisfactory – 3	**PASS**
6 – good 7 – very good	Good – 4	
8 – almost excellent 9 – excellent 10 – perfect	Excellent – 5	

7. Course description.

Topic 1: Formal systems. Syntax and semantics.

Topic 2: Predicate logic as a specification language for AI and CS.

Discuss several subtle examples of encoding problems into predicate logic. Discuss the importance of distinguishing syntax and semantics. Formally describe the syntax of predicate logic. Formally define the domain of interpretation of predicate logic (universe and interpretation of symbols). Informally describe the semantics of predicate logic formulas, including connectives and quantifiers.

Topic 3: Models. Semantics of predicate logic. Finite and infinite models.
Recap on the formal syntax of predicate logic. Bound and free variables. Formal semantics of predicate logic.Validity, satisifiability, etc. Brief introduction to reasoning about predicate logic formulas.

Topic 4: Automatized provers for predicate logic: Soundness and completeness.
The unfolded tree analysis in predicate logic. A finite system of inference rules. Soundness and completeness. Bound and free variables.

Topic 5: Predicate logic: Undecidability. Decidable fragments.
Simulation Turing/Minsky machines within predicate logic. Decidable procedures for propositional logic. Decidable procedures for pure monadic predicate logic.

Topic 6: Hoare triples as a language to specify the dynamic behaviour of programs/plans in AI and CS.
Introduction to Hoare logic as the foundation of many successful techniques for formal verification of software. Provide intuition about program correctness through examples of program specifications in Hoare logic. Formally define the syntax and semantics of Hoare triples. Distinguish between the notion of partial and total correctness. The importance of auxiliary variables. Use Hoare Triples

Topic 7: Hoare logic. Inference rules. Soundness and Completeness. Automatized Inference of Loop Invariants.
Formally define inference rules of Hoare logic for reasoning about partial program correctness. Explain the connection between forwards and backwards axiom for assignment and weakest preconditions for assignment. Explain how reasoning about predicate logic formulas is used for verifying programs using Hoare Logic. Apply the Hoare Logic inference rules to prove (or disprove) correctness of loop free programs. Use loop invariants to prove correctness of programs with loops. Discuss the difference between inference rules for partial and total correctness. Discuss extensions of Hoare Logic to support other programming language features, such as *pointers*. Classic (single-path) symbolic execution. Recap: Syntax and semantics of while programs. Proving Hoare triples of programs using forward symbolic execution. Understand an algorithm for generating loop invariants automatically.

Topic 8: SAT Solvers. Symbolic model checking. Horn clauses and functional dependencies in relational databases. The DPLL algorithm and its improvements.
What is SAT. Polynomial-time algorithms in the case of Horn clauses and functional dependencies in relational databases. Normal forms for general case. Tseitin transformation. Understand the DPLL algorithm: unit propagation, pure literal elimination. Conflict-Driven Clause Learning as an efficient tool for solving the Boolean satisfiability problem in practice.

8. Educational technologies

The standard structure of the course, "lectures + tutorials", will be supported by a system of homework assignments and reading material handed out on a weekly basis.

The model solutions will be discussed in class and posted on the Web. Postgraduate students are strongly encouraged to take part in these discussions.

On top of that, the handouts will be regularly updated to meet postgraduate students' requests.

As a result, each of the postgraduate students will collect the whole supporting package including handouts, practice problems, model answers to the homework assignments and midterms, etc.

9. Reading and Materials

Literature:

1. M.Huth and M.Ryan. Logic In Computer Science. Cambridge University Press.

Additional literature:

1. Новиков П.С. Элементы математической логики. 2-е изд. М.:Наука, 1973
2. Н. Верещагин, А. Шень, Языки исчисления, МЦНМО, 2012.
3. Nipkow, T., Grumberg, O., and Hauptmann, B., eds. Software Safety and Security: Tools for Analysis and Verification. Amsterdam, NLD: IOS Press, 2012.
4. Hoare, C. A. R. (October 1969). An axiomatic basis for computer programming. Communications of the ACM 12 (10): 576-580. doi:10.1145/363235.363259.
5. Michael R. Garey and David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and co., New York.
6. Carla P. Gomes, Henry Kautz, Ashish Sabharwal, Bart Selman (2008). "Satisfiability Solvers." In Frank Van Harmelen, Vladimir Lifschitz, Bruce Porter. Handbook of knowledge representation. Elsevier. pp. 89-134. doi:10.1016/S1574-6526(07)03002-7

Literature for self-study:

1. George S. Boolos, John P. Burgess, and Richard C. Jeffrey. Computability and logic. Fourth edition. Cambridge University Press, Cambridge, 2002.
2. John C. Reynolds (2009). Theory of Programming Languages. Cambridge University Press.

10. Equipment.

Sufficient PC quantity for students.