Government of Russian Federation
Federal State Autonomous Educational Institution
of High Professional Education
National Research University "Higher School of Economics"
Faculty of Computer Science
School of Data Analysis and Artificial Intelligence

Syllabus for the course

"Fundamentals of Knowledge Representation"

"Основы представления знаний"

для направления 09.06.01 "Информатика и вычислительная техника," профили 05.13.01 "Системный анализ, управление и обработка информации," 05.13.11 "Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей," 05.13.17 "Теоретические основы информатики," 05.13.18 "Математическое моделирование, численные методы и комплексы программ"
подготовки научно-педагогических кадров в аспирантуре.

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

Approved on the meeting of the Academic Council of
the PhD School of Computer Science.

_____
___ _____ 2017

Moscow, 2017

"Fundamentals of Knowledge Representation"

"Основы представления знаний"

Course Overview

1 The aim of the course

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 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 English**, with "subtitles" **in Russian**, 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.

2 Prerequisite, Reading Material

A decent knowledge of logic and math is needed.

In fact, the course is *self-contained* - all basic concepts will be introduced step-by-step in class.

The pace of the course is driven not only by the syllabus but by the lecturer-students interaction.

Recommended Reading:

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

The following can be recommended to provide the additional advanced information.

(1) Новиков П.С. Элементы математической логики. 2-ое изд. М.: Наука, 1973

(2) George S. Boolos, John P. Burgess, and Richard C. Jeffrey. Computability and logic. Fourth edition. Cambridge University Press, Cambridge, 2002.

(3) Николай Верещагин, Александр Шень. Языки и исчисления, МЦНМО, 2012 г.

(4) Nipkow, T., Grumberg, O., and Hauptmann, B., eds. Software Safety and Security: Tools for Analysis and Verification. Amsterdam, NLD: IOS Press, 2012.

(5) 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.

(6) John C. Reynolds (2009). Theory of Programming Languages. Cambridge University Press.

(7) 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, 1979.

(8) Carla P. Gomes, Henry Kautz, Ashish Sabharwal, Bart Selman (2008). "Satisfiability Solvers." In Frank Van Harmelen, Vladimir Lifschitz, Bruce Porter. Handbook of knowledge representation. Foundations of Artificial Intelligence 3. Elsevier. pp. 89-134. doi:10.1016/S1574-6526(07)03002-7

Since the course is intended to follow the most recent achievements, there is no textbook which would serve the course in full extent.

Therefore, the standard structure of the course is "lectures + tests". The model solutions will be discussed in class and posted on the Web. Students are strongly encouraged to take part in these discussions.

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

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

3 Assessment

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

(ii) Final exam worth 60% of the course marks.

4 Learning outcomes

The content is deliberately planned around the concept of learning outcomes or objectives. So that a student can garner some understanding of what the course should allow them to learn and that they should check (in a whatever way seems appropriate) after studying that material that they have achieved and can demonstrate those outcomes.

- Understand the aim of the course.
- Get a big picture on the materials to be covered by the course.
- 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.
- Prove or disprove predicate formulas by means of the unfolded tree analysis, *a kind of a theorem prover*
- Prove soundness and completeness for some finite system of inference rules.
- Simulate computations within predicate logic.
- Decidable procedures for propositional logic.
- Decidable procedures for pure monadic predicate logic.
- Specify programs formally using Hoare triples
- Manually run symbolic execution rules for loop-free programs
- Use symbolic execution rules to prove loop-free programs semi-automatically
- Understand the definition of loop invariants
- Write loop invariants for simple arithmetic programs
- Prove simple arithmetic programs using loop invariants
- Understand symbolic execution for loops
- Understand an algorithm for inferring loop invariants automatically
- Apply the algorithm for simple arithmetic programs
- Understand SAT solvers

- Understand the DPLL algorithm:
- Apply Conflict-Driven Clause Learning as an efficient tool for SAT solving in practice.

5 Outline

(1) *Topic 1*: **Formal systems. Syntax and semantics.**

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

Learning outcomes:
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.

(3) *Topic 3*: **Models. Semantics of predicate logic. Finite and infinite models.**

Learning outcomes:
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.

(4) *Topic 4*: **Automatized provers for predicate logic: Soundness and completeness.**

Learning outcomes:
The unfolded tree analysis in predicate logic.
A finite system of inference rules. Soundness and completeness. Bound and free variables.

(5) *Topic 5*: **Predicate logic: Undecidability. Decidable fragments.**

Learning outcomes:
Simulation Turing/Minsky machines within predicate logic.

Decidable procedures for propositional logic.
Decidable procedures for pure monadic predicate logic.

(6) *Topic 6*: **Hoare triples as a language to specify the dynamic behaviour of programs/plans in AI and CS.**

Learning outcomes:
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 to specify simple program in the While programming language.

(7) *Topic 7*: **Hoare logic. Inference rules. Soundness and Completeness. Automatized Inference of Loop Invariants.**

Learning outcomes:
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.

(8) *Topic 8*: **SAT Solvers. Symbolic model checking. Horn clauses and functional dependencies in relational databases. The DPLL algorithm and its improvements.**

Learning outcomes:
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.

6 Schedule

N	Topic	Total	In class		Self-study
			Lectures	Seminars	
1	Formal systems. Syntax and semantics.	19	2	2	15
2	Predicate logic as a language for CS	19	2	2	15
3	Models. Semantics of predicate logic.	19	2	2	15
4	Automatized provers for predicate logic.	19	2	2	15
5	Predicate logic: Undecidability. Decidable fragments.	19	2	2	15
6	Hoare triples as a language within a dynamic context.	19	2	2	15
7	Hoare Logic. Inference Rules.	19	2	2	15
8	SAT solvers. Complexity issues.	19	2	2	15
	Total	152	16	16	120