Syllabus

1. Course Description
    a. Title of a Course: **Automatic proof checking and algorithm verification**
    b. Pre-requisites: basic knowledge of logic; familiarity with lambda calculi and/or functional programming is not presupposed.
    c. Course Type: elective
    d. Abstract: Modern math proofs are sometimes very long and sophisticated, so a human expert cannot full analyse their details and insure correctness. An analogous issue comes from IT: big projects usually contain bugs, and testing is not always capable of finding all of them. A more reliable method is verification: formal proof of the correctness statement of a program w.r.t. a formally given specification. Finally, there are mixed cases, when a mathematical proof is a result of brute-force case analysis performed on a computer (classic example: the 4-colour problem). In this situation one needs to formally verify the program in order to finalise the proof. The course aims to discuss formalization on mathematical proofs in one of the proof assistants, Coq (http://coq.inria.fr/). The course also includes necessary material from mathematical logic and theoretical computer science: intuitionistic logic, lambda calculus, elements of type theory, Curry – Howard correspondence, elements of functional programming. Along with theoretical lectures, the course also includes practical classes.
2. Learning Objectives: to familiarize students with the basic ideas of proving theorems in constructive logic, and formalizing these proofs in Coq.
3. Learning Outcomes:
    - Know principles of intuitionistic logic, its connections to dependent type theory, and the Calculus of Inductive Constructions.
    - Be able to formalize elementary mathematical proofs in the Coq proof assistant.
    - Be able to extract working functional code from constructive proofs of existential claims in Coq, and also to verify existing functional code.
4. Course Plan:
    - Provers and Proof Assistants. Introduction to Coq. The CoqIDE integrated interactive environment.
    - Intuitionistic propositional logic. Hilbert-style and natural deduction calculi. Simply typed lambda calculus and Curry – Howard correspondence (formulae = types, proofs = terms)
    - Kripke models for intuitionistic propositional logic. Soundness and completeness, disjunction property.

- Dependent types: dependent product and sum as Curry – Howardesque correspondents of quantifiers (in higher-order logic). Elements of type theory. The Calculus of Constructions (CoC)
- Recursive (inductive) type definitions. The Calculus of Inductive Constructions (CIC). Example: natural numbers (type 'nat'); proving facts from elementary arithmetic in Coq. The Gallina functional language.
- Beta-reduction in lambda-calculi and its properties Church-Rosser(CR), weak normalization (WN), strong normalization (SN). Counter-examples. WN for Gallina
- Extraction of functional code from constructive proofs in Coq. Translating between functional programming languages.Verification existing Haskell code via hs-to-coq

5. Reading list

a. Required

1. Optional Coq Reference Manual: https://coq.inria.fr/distrib/current/refman/
2. Coq Standard Library Reference: https://coq.inria.fr/distrib/current/stdlib/
3. E. Giménez, P. Castéran. A Tutorial on [Co-]Inductive types in Coq. http://www.labri.fr/perso/casteran/RecTutorial.pdf
4. D. van Dalen. Logic and Structure. Springer, 2013.

b. Optional

1. B. C. Peirce *et al.* Logical Foundations (vol. 1) https://softwarefoundations.cis.upenn.edu/lf-current/index.html
2. D. Gabbay, V. Shehtman, D. Skvortsov. Quantification in Nonclassical Logic, vol. 1. Draft of the 2nd edition: http://lpcs.math.msu.su/~shehtman/n.ps
3. V. Krupski, S. Kuznetsov. Practicum in Mathematical Logic. COQ [in Russian]. Moscow State University, 2013. http://www.mi-ras.ru/~sk/lehre/coq/coq_pract.pdf

6. Grading System

Final Grade = 60% Cumulative Grade + 40% Final Exam

The Cumulative Grade (from 1 to 10) is computed from several home assignments (both practical, in Coq, and theoretical, in written form), handed out during the semester, provided their solutions are submitted on time. (Grading of solutions submitted after deadlines is at the discretion of the instructor.)

The Final Exam is a theoretical written exam in the end of the semester, which could include both theoretical questions and problem solving. The Final Exam is graded from 1 to 10.

Rounding is performed to the nearest integer; $x.5$ is rounded to $x+1$.

Any fact of cheating results in a **zero** grade.

7. Guidelines for Knowledge Assessment

   Since Coq automatically checks proofs, assessment of practical tasks can be performed on an automated basis. However, instructors are encouraged to discuss code with the students, in order to avoid badly written or inoptimal code. Written theoretical assignments and the Final Exam are to be assessed as usual mathematical written works.

8. Methods of Instruction

   Lectures are delivered in a classical style, extended with demonstrations in Coq. Seminars and practical classes include both discussions of theoretical problems and programming in Coq.

9. Special Equipment and Software Support (if required)

   A computer class with Coq/CoqIDE installed is highly recommended for seminars/practical classes. For lectures, there should be one computer with Coq/CoqIDE installed, for the lecturer to perform demonstrations. Coq/CoqIDE is free software (LGPL license), available for all major operating environments, downloadable from http://coq.inria.fr/