Computational Logic

http://cs.ubbcluj.ro/~lupea/LOGICA/Engleza

Objectives:

The aim of the course is the presentation of the *logical foundations of computer science*: propositional calculus and predicate calculus, theorem proving methods, Boolean algebras and Boolean functions. The connection with logic programming and logic circuits is presented. Additionally, notions related to *information representation* are introduced.

Competencies:

- **to use numeration systems and internal numbers representations;**
- **to understand the theoretical aspects of classical logics;**
- **to understand classical logics from a computational perspective (apply specific proof methods)**
- **to model the human reasoning and mathematical reasoning using propositional logic and predicate logic;**
- **to understand logic circuits and to simplify them using specific simplification methods for Boolean functions.**

Methods:

- **lectures, exercises, individual study;**
- **specific bibliographic materials are used (books, articles, Internet Resources).**
- **each student, individually, has to solve and present during the seminars, problems from an existing list of proposed problems (M. Lupea, A. Mihis , 2015).**
- **optional homework**

Content:

I. Numeration systems, numbers representations

II. Classical logics: propositional and predicate logics

III. Boolean algebras, Boolean functions and logic circuits

I. Numeration systems, information encoding, numbers representations

====> prerequisite for the discipline: *Computer Architecture*

- **Numeration systems:**
  1. **definitions, representation and operations (algorithms for comparison, addition, subtraction, multiplication, division by a digit) of numbers in a base *b*.**
  2. **conversions between two bases using three methods for integer and rational numbers.**
  3. **examples for particular bases: 2,3,4,6,8,10,16.**
- **Numbers representation**
  1. **representation for unsigned integers, operations (!!overflow!!), algorithms for multiplication and division.**
  2. **representation for signed integers: direct code, inverse code, complementary code, operations and subunitary convention.**
  3. **representations for real numbers: fixed-point representation, floating-point representation.**

References:

1) **F. Boian: *Bazele Matematice ale Calculatoarelor*, Editura Presa Universitara Clujeana, 2002.**
2) **M. Cocan, B. Pop: *Bazele matematice ale sistemelor de calcul* (chapter 1), Editura Albastra, Cluj-Napoca, 2001.**
3) **A. Vancea, F. Boian, D.Bufnea, A.Gog, A.Darabant, A.Sabau: *Arhitectura calculatoarelor*. *Limbajul de asamblare 80x86*, (chapter 1), Editura Risoprint, Cluj-Napoca, 2005.**

II. Classical logics: propositional logic and predicate logic

***Propositional calculus* and *predicate calculus* are presented from an algebraic point of view and as deductive systems (computational perspective). Theorem proving methods are used to decide if a statement (*conjecture*) is a logical consequence of a set of statements (*axioms* and *hypotheses*).**

***Aim:* formalization of human and mathematical reasoning using these classical logics.**
===> *prerequisite for the disciplines:* Logic programming, Artificial Intelligence, Automated theorem proving systems.

1. **Syntax: connectives, quantifiers, terms, atoms, formula, clause, literal, formulas.**
2. **The formal (axiomatic) system associated to propositional/predicate logic.**
3. **Semantics of propositional/predicate logic: interpretation, model, valid formula, consistent formula, inconsistent formula, logical consequence.**
4. **Normal forms in propositional and predicate logics.**
5. **Theorem of soundness and theorem of completeness for propositional/predicate logic; noncontradiction, coherence and decidability/semi-decidability of propositional/predicate logic.**
6. **Semantic tableaux method – a semantic and refutation proof method.**
7. **Resolution method – a direct and refutation proof method. Refinements of resolution.**
8. **Formalization of common-sense and mathematical reasoning using propositional and predicate logic.**

Timeline of research in logic

450 B.C. Stoics - propositional logic (PL), inference

322 B.C. Aristotle -``syllogisms“ (inference rules),quantifiers

1565 Cardano - probability theory (PL + uncertainty)

1646 -1716 Leibniz research for a general decision procedure
to check the validty of formulas

1847 Boole – algebras, formalization of propositional logic

1879 Gottlob Frege – predicate or first-order logic (FOL)

1889 Peano - 9 axioms for natural numbers

1920 Hilbert‘s program

1922 Wittgenstein - proof by truth tables

1929 Gödel completeness theorem of FOL

1930 Herbrand -a proof procedure for FOL based on propositionalization

1931 Gödel incompleteness theorems for the consistency of Peano axioms

1936 Gentzen a proof for the consistency of Peano axioms in set theory

1936 Church and Turing: undecidability of FOL

1954 Davis First machine-generated proof

1955 Beth- Semantic Tableaux

1957 Newell, First machine-generated proof in Logic Calculus

1957 Kangar, Lazy -substitution by free (dummy) Vars Prawitz

1958 Prawitz - First prover for FOL

1959 Gilmore, Wang- more provers

1960 Davis-Putnam Procedure

1963 Robinson Unification, resolution

1968 R.Smullyan – Semantic tableaux proof method

Types of logics:

- **classical logics: propositional and predicate logics**
- **modal and temporal logics**
- **multivalued and fuzzy logics**
- **nonmonotonic logics**

Automated Theorem Proving **deals with the development of computer programs which show that some statement (the *conjecture*) is a *logical consequence* of a set of statements (the *axioms* and the *hypotheses*).**

Results obtained in *mathematics* using ATP systems:

- **the TP system *EQP*. In 1933 Herbert Robbins conjectured that a particular group of axioms forms a basis for Boolean algebra, but neither he nor anyone else could prove this. The proof was found in October 10, 1996, after about 8 days of search by EQP.**
- **the TP system *Otter* - used to prove several results in quasi-groups.**
- **the geometry prover: *Geometry Expert* has been used to obtain new results in Euclidean geometry.**

Software generation **is an economical important real world application of ATP.**

- **The *KIDS* system developed at Kestrel Institute has been used to derive scheduling algorithms that have outperformed currently used algorithms. KIDS provides intuitive, high level operations for transformational development of programs from specifications.**
- **The *AMPHION* project, sponsored by NASA, is used to determine appropriate subroutines combined to produce programs for satellite guidance. By encapsulating usable functionality in software components, and then reusing those components, AMPHION can develop software in less time than human programmers.**

Software verification **is an obvious and attractive goal, which is slowly being realized using TP technology.**

- **The *Karlsruhe Interactive Verifier (KIV)- verify a large* range of software applications. These include some case studies of academic software, e.g., implementation of set functions, tree and graph representation and manipulation. KIV is also use for *industrial application,* with pilot studies undertaken in various domains, including a software controlled railway switch, safe command transfer in a space vehicle, and supervision of neutron flow in a nuclear reactor.**

- ***PVS*** **is a verification system that has been used in various applications, including diagnosis and scheduling algorithms for fault tolerant architectures, and requirements specification for portions of the space shuttle flight control system. NASA uses ATP to certify safety properties of aerospace software that has been automatically generated from high-level specifications. Their code generator produces safety obligations that are provable only if the code is safe. An ATP system discharges these obligations, and the output proofs, which can be verified by an independent proof checker, serve as certificates.**

Hardware verification **is the largest industrial application of ATP. IBM, Intel, and Motorola are among the companies that employ ATP technology for verification.**

- *ACL2* **system - used to obtain a proof of the correctness of the floating point divide code for AMD's PENTIUM-like AMD5K86 microprocessor**
- *ANALYTICA* **- used to verify a division circuit that implements the floating point standard of IEEE.**
- *HOL* **system - used at Bell Laboratories for hardware verification.**

Potential fields**: biology , social science, medicine, commerce, etc.**

Examples of dedicated (educational) automated theorem provers:

- **based on semantic tableaux method: 3TAP, pTAP, leanTAP, Cassandra;**
- **based on resolution method: OTTER, PCPROVE, AMPHION, Jape;**
- **based on semantic trees + Herbrand theorem : HERBY;**
- **based on model elimination calulus: SETHEO;**

Implementations: PROLOG, LISP, C/C++,…

Propositional logic

Logical propositions are models of propositional assertions from natural language, which can be *true* or *false*.

P: It is sunny. **Q: It is hot outside.** **R: I go to the swimming pool.**

S: *If* it is sunny *and* it is hot outside *then* I will go to the swimming pool.

S: $P \wedge Q \rightarrow R$

"Theorem proving": **From P,Q,S (the hypotheses) can we deduce (infer) R (the conjecture)?**

Predicate (first-order) logic

The axioms that defines the natural numbers:

a1. *Every* natural number has exactly one immediate successor.

existence: $(\forall x)(\exists y) equal(y, succesor(x))$

unique: $(\forall x)(\forall y)(\forall z)(equal(y, succesor(x)) \wedge equal(z, successor(x)) \rightarrow equal(y, z))$

a2. The natural number 0 is *not* the immediate successor of any natural number.

$\neg(\exists x) equal(0, succesor(x))$

a3. *Every* natural number except 0 has exactly one immediate predecessor.

$(\forall x)(\exists y)(\neg equal(0, x) \wedge equal(y, predecessor(x)))$

Functions: *successor*, *predecessor* **; Predicates:** *equal*

Reasoning modeling using predicate logic:

Hypotheses:

H1. If x is perpendicular to y then x intersects y.
H2. If x is parallel to y then x doesn't intersect y.
H3. If x is perpendicular to y and z is perpendicular to y then x is parallel to z.
H4. d1 is perpendicular to d.
H5. d is perpendicular to d2.

Conclusion:C. d1 doesn't intersect d2.

Check whether the conclusion C is derivable from the set of hypotheses.

Succession to the British throne – common-sense human reasoning

Hypotheses:

H1: If *x* is the king and *y* is his oldest son, then *y* can become the king.
H2: If *x* is the king and *y* defeats *x*, then *y* will become the king.
H3: *RichardIII* is the king.
H4: *HenryVII* defeated *RichardIII*.
H5: *HenryVIII* is *HenryVII*'s oldest son.

Conclusion C: Can *HenryVIII* become the king?

Check whether the conclusion C is derivable from the set of hypotheses

References

1. **M. Ben-Ari: *Mathematical Logic for Computer Science*, Ed. Springer, 2001.**
2. **W. Bibel: *Automated theorem proving*, View Verlag, 1987.**
3. **C.L. Chang, R.C.T. Lee: *Symbolic Logic and Mechanical Theorem Proving*, Academic Press 1973.**
4. **M. Clarke: *Logic for Computer Science*, Ed. Eddison-Wesley 1990.**
5. **J.P. Delahaye: *Outils logiques pour l'intelligence artificielle*, Ed Eyrolls, 1986.**
6. **M. Fitting: *First-order logic and Automated Theorem Proving*, Ed. Springer Verlag, 1990.**
7. **Mihaela Malita, Mircea Malita: *Bazele Inteligentei Artificiale, Vol. I, Logici propozitionale*, Ed. Tehnica, Bucuresti, 1987 – library.**
8. **M. Lupea, A. Mihiş: *Logici clasice şi circuite logice. Teorie şi exemple*, Editura Albastra, edition I-2008, edition II – 2009, edition III-2011.**
9. **M. Lupea, A. Mihis: *A Computational Approach to Classical Logics and Circuits*, Editura Risoprint, Cluj-Napoca, 2015.**
10. **L.C. Paulson: *Logic and Proof*, Univ. Cambridge, 2000, on-line course.**
11. **M. Possega: *Deduction Systems*, Inst. of Informatics, 2002, on-line course.**
12. **(ed) A.Thayse: *From standard logic to Logic Programming*, Ed. J.Wiley, vol1(1989), vol2(1989), vol3(1990).**
13. **D.Tatar: *Bazele matematice ale calculatoarelor*, edition 1999- library.**
14. **D.Tatar: *Inteligenta artificiala: demonstrare automata de teoreme si NLP*, Ed. Microinformatica, 2001 – library.**

III. Boolean algebras, Boolean functions and Logic circuits

1. ***Boolean algebras*: definition, properties, principle of duality, example.**
2. ***Boolean functions*: definitions, maxterms, minterms, the canonic disjunctive form and the canonic conjunctive form, transformation.**
3. ***Simplification of Boolean functions*:**

- **definitions: maximal monoms, central monoms, factorization;**
- **Veitch-Karnaugh diagrams method – for functions of 2-3 variables;**
- **Quine's method, Moisil's method.**

4. ***Logic circuits***

- **definitions, representations for basic gates (“and”, “or”, “not”) and derived gates (“xor”, “nand”, “nor”) and relations between them.**
- **examples of logic circuits: adder, subtractor, decoder, encoder, comparator .**

References:

1. **M. Cocan, B. Pop: Bazele matematice ale sistemelor de calcul (chapter 2), Editura Albastra, Cluj-Napoca, 2001 – library.**
2. **M. Lupea, A. Mihiş: *Logici clasice şi circuite logice. Teorie şi exemple.*, Editura Albastra, edition I -2008, edition II - 2009, edition III - 2011.**
3. **M. Lupea, A. Mihis: *A Computational Approach to Classical Logics and Circuits,* Editura Risoprint, Cluj-Napoca, 2015.**
4. **D.Tatar: Bazele matematice ale calculatoarelor, edition 1999- library.**

Evaluation - shares in the final grade (%)

1. **Written paper** (seminar 5 -one hour) with subjects from the first part: **15%**
   - operations and conversions in different numeration bases
   - internal representations of integer and real numbers
2. Final written paper with 3 subjects (theory and exercises) from courses and seminars 3-14: **60%**
   - propositional logic
   - predicate logic
   - Boolean algebras, Boolean functions and their simplification, logic circuits
3. Seminars' activity: **20%**
   - **attendance to seminars is mandatory for at least 75% - 11 seminars.**
   - responses + individual presentations of exercises from an existing list of proposed exercises.

4. Optional homework for **10**% (increases the final grade):
   - an application (programming languages: Pascal, C, C++, Java,…) related to conversions and operations in different numeration systems;
     - deadline for application submission: the second week of December 2015.

   or

   - **t**o solve a set of exercises which model the mathematical reasoning and the common-sense reasoning, using the studied proof methods for propositional and predicate logics.
     - deadline for homework submission: the first week of January 2016.

   !! Written papers and seminar activity: at least grade 5.

   !! Attendance to courses increases the final grade.

Organization, exceptional situations

The official statutes of the university regarding the students' attendance to the didactic activities and plagiarism, hold.