Autumn 2017
F17LP Logic and Proof

Lecturer: Prof Mark V Lawson
Office: CM F12
email: m.v.lawson@hw.ac.uk

Aims This is an introduction to first order logic suitable for first and second year mathematicians and computer scientists. There are three components to this course: proofs, propositional logic and predicate logic. Proofs are the basis of mathematics — how do we know what we say is true? — and also of computer science — how do I know this program will do what I think it will do? Propositional logic and predicate logic are the two ingredients of first-order logic and are what make proofs work. Propositional logic deals with proofs that can be analysed in terms of the words *and, or, not, implies* whereas predicate logic extends this to encompass the use of the words *there exists* and *for all.*

How much maths? Surprisingly little school-type mathematics is needed to learn and understand logic: this course doesn't involve any calculus, for example. In fact, most of the maths you studied at school was invented before 1800 and so is (mainly) irrelevant to the needs of computer science (a slight exaggeration, but not by much). The real mathematical prerequisite is an ability to manipulate symbols: in other words, basic algebra. Anyone who can write programs should have this ability already.

Tutorials From week 2 onwards, there will be one tute a week where you can ask general questions and work through the exercise sheets. I shall organize the tutes in the first lecture.

Assessment There will be two take-home tests each worth 10% and one final 2 hour exam worth 80%. Last year's exam paper and solutions are available on the course website.

Material I have set up a website for the course that can be accessed directly via my homepage `http://www.macs.hw.ac.uk/~markl/teaching/LOGIC/` Alternatively, you can access the same information via VISION. During the semester, I shall post up lecture notes as well as exercise sheets and their solutions.

Syllabus

1. Propositional logic (PL)

- *Informal propositional logic*: I shall introduce (PL) in the first instance as a very simple kind of language in which to express various kinds of simple decision-making scenarios. Later we shall see that it can also be viewed as a programming language.

- *Syntax of propositional logic*: Well-formed formulae (wff), atoms, compound statements, trees (examples of data structures), parse trees, principal connective, order of precedence rules and brackets.

- *Semantics of propositional logic*: Definition of the connectives by means of truth tables, truth assignments, truth tables in general, contradictions, contingencies and tautologies, satisfiability and the satisfiability problem (SAT).

- *Logical equivalence*: Logical equivalence ($\equiv$), some important examples of logical equivalences (will reappear in disguise when we study Boolean algebras), how to say ‘exactly one’.

- *An example*: The satisfiability problem: example using sudoku.

- *Adequate sets of connectives*: Examples of adequate sets of connectives, **nand** and **nor**; truth functions, theorem: every truth function is the truth table of some wff.

- *Normal forms*: Negation normal form (NNF), literals, disjunctive normal form (DNF), truth functions, conjunctive normal form (CNF), Horn formulae.

- *$P = NP$?*: The satisfiability problem, $P = NP$? and its significance. This section is mainly for background knowledge and interest and to explain how (PL) can also be viewed as a programming language.

- *Valid arguments*: What do we mean by an argument? The use of (PL) in formalizing certain kinds of simple arguments — one of the goals of logic: to mechanize reasoning. The semantic turnstile $\models$ and its properties; classical arguments: modus ponens, modus tollens, disjunctive syllogism, hypothetical syllogism (these terms do not have to be memorized).

- *Truth trees*: Derivation of truth tree rules, the truth tree algorithm, using truth trees to solve problems: determining whether a wff is a tautology, determining whether an argument is valid, determining whether a finite set of wff is satisfiable, writing a wff in DNF; the symbol $\vdash$.

2. **Boolean algebras (BA)**

   - *Set theory*: Introduction to sets, the power set of a set, intersections, unions, and relative complements, Venn diagrams. Simplifying Boolean expressions; how to organize proofs in Boolean algebras.
   - *Definition of Boolean algebras*: Definition motivated by logical equivalences in PL, the Lindenbaum algebra of PL, simple properties of Boolean algebras, the two-element Boolean algebra.
   - *Binary arithmetic*: Converting between base 10 and base 2; adding two numbers in base 2.
   - *Circuit design*: Shannon's work on switching circuits and Boolean algebra, logic gates, theorem: every combinational circuit can be constructed from basic logical gates; how to build an adding machine.
   - *Transistors*: Theorem: every combinational circuit can be constructed from transistors.

3. **First-order logic (FOL)**

   - *Splitting the atom: names, predicates and relations*: Names or constants and variables; 1-place predicates, n-place or n-ary predicates, binary and ternary predicates, the arity of a predicate; atomic formulae; relations of different arities; directed graphs and binary relations.
   - *Structures*: Domains and relations.
   - *Quantification $\forall$ and $\exists$* : $\forall$ regarded as an infinite conjunction and $\exists$ regarded as an infinite disjunction; a famous argument on Socrates' mortality analysed.
   - *Syntax*: A first-order language, formulae, parse trees; subtrees of trees, free and bound variables, the scope of a quantifier, closed formula/sentence.
   - *Semantics*: Interpretations, models, logically valid sentences; valid arguments; logical equivalence; satisfiable; contradictions.
   - *De Morgan's laws for $\forall$ and $\exists$*: $\neg(\forall x)A \equiv (\exists)\neg A$ and $\neg(\exists x)A \equiv (\forall x)\neg A$.
   - *Truth-trees for first-order logic*: The two additional rules needed to deal with $(\forall x)$ and $(\exists x)$; examples.
   - *The Entscheidungsproblem*: An informal and non-examinable description of Turing's work.

Books There are two kinds of books: those you read and those you work from. For background reading during the long Scottish winter nights, I recommend [1, 3, 4]. The remaining books are work books. Of those, you might want to start with Zegarelli [8]; don't be put off by the *Dummies* tag, since it isn't a bad introduction to logic for CS students — who are most certainly not, on the whole, dummies. For some of the more obviously mathematical content, Lipschutz and Lipson [5] is useful. The chapters you want, in the second edition are: 1, 2, 3, 4, and 15. However, let me emphasize that we won't be covering everything you will find there. Another maths type book is Hammack [2]. This might be more suitable for the more mathematically inclined student. The book by Smullyan [6] I used when I was writing this course. The book by Teller [7] is a step-up from the Dummies book.

References

[1] A. Doxiadis, C. H. Papadimitriou, *Logicomix*, Bloomsbury, 2009.

[2] R. Hammack, *Book of proof*, VCU Mathematics Textbook Series, 2009. This book can be downloaded for free from `http://www.people.vcu.edu/~rhammack/BookOfProof/index.html` or you can also find it on the webpage for this course.

[3] D. Harel, *Computers Ltd What they really can't do*, OUP, 2012.

[4] D. R. Hofstadter, *Gödel, Escher, Bach: an eternal golden braid*, Basic Books, 1999.

[5] S. Lipschutz, M. Lipson, *Discrete mathematics*, second edition, McGraw-Hill, 1997.

[6] R. M. Smullyan, *First-order logic*, Dover, 1995.

[7] P. Teller, *A modern formal logic primer*, Prentice Hall, 1989. This book can be downloaded for free from `http://www.ma.hw.ac.uk/~markl/teaching/LOGIC/Teller.html` or you can also find it on the webpage for this course.

[8] M. Zegarelli, *Logic for dummies*, Wiley Publishing, 2007.