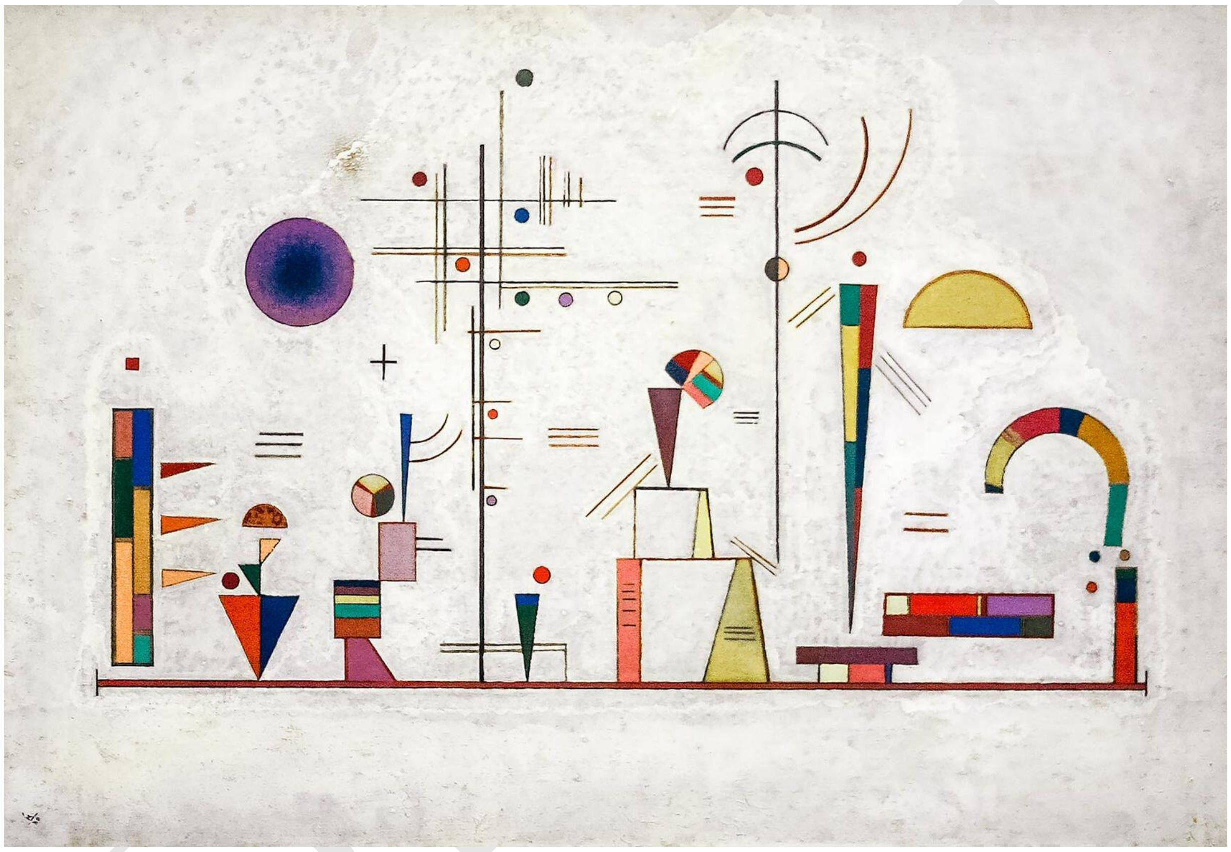
Mathematical Logic through Python



Yannai A. Gonczarowski   Noam Nisan

Draft; comments welcome

Mathematical Logic through Python

Yannai A. Gonczarowski Noam Nisan

Draft; comments welcome

To Eshed, whose syntax and semantics logically evolved while this book did

Y.A.G.

To Michal, logically and illogically

N.N.

Contents

Preface

Mathematical Logic 101 is a beautiful course. Gödel's Theorems are arguably the most profound and deep truths taught throughout the entire undergrad theoretical curriculum. Nonetheless, it seems that among many computer science and engineering students this course suffers from the reputation of being an unintelligible course full of technical, unin-sightful proofs. Students lose themselves in endless inductions, and do not fully understand what it means, e.g., to "prove that anything that is true can be proven." Indeed, how can this not be confusing when the two occurrences of "prove" in that sentence have two distinct meanings, the latter referring to a precise very strict mathematical "proof" object that is defined during this course while the former refers to the free-text proofs that we have been taught since our first year of undergrad? This book drastically re-envisions the Mathematical Logic 101 course, conveying the same material but tapping into the strengths of the ever-growing cohort of programming-oriented students to do so.

How does one help programming-oriented students to not lose themselves among endless little details in proofs, losing sight of the overarching message of the course? We set out to make this course less abstract, more intuitive, and maybe even exciting, by tapping into the context where such students are used to comfortably deal with endless little details on the way to a larger goal without ever missing the forest for the trees: computer programming. We redesigned the entirety of this very theoretical course from scratch to be based upon a series of programming exercises, each corresponding either to a theorem/lemma/corollary, or to a step toward such.

For example, the main result of the first half of a standard Mathematical Logic 101 course is the "Tautology Theorem" (a variant of the Completeness Theorem for propositional logic), which asserts that every tautology—every statement that holds in every possible model or setting—can be proven to hold using a small set of axioms. The corresponding programming exercise in this book is to write a function (based on functions from previous exercises, of course) whose input is a formula (an object of class `Formula`, which the students implement in a previous exercise) and whose output is either a model in which this formula does not hold (that is, a counterexample to the formula being a tautology) or a proof (an object of class `Proof`, which the students implement in a previous exercise) of this formula. Obviously, whoever can write such a function, including all its recursively implemented helper functions, completely understands the reasoning in the proof of the Tautology Theorem, including all its inductively proven lemmas. (And this holds even more so for students who naturally grasp recursive code far better than they do inductive proofs.) In our experience, students with a background in programming for the most part even understand this proof better having actively coded its functionality themselves than had they only passively seen the proof being written on the blackboard by a teacher. In the process of moving from proving to programming, we have in fact also disambiguated the two meanings of "prove" in the above statement of "prove that whatever is true can be proven": we transformed the former "prove" into "program in code" and the latter "can be proven" into "is the corollary of a valid `Proof` object." This disambiguation by way of defamiliarization of each of these occurrences of "prove" achieves great clarity, and furthermore allows the students to more easily reexamine their intuitions and unconscious assumptions about proofs.

This book evolved from a course that we have been teaching at the Hebrew University of Jerusalem since 2017, first as an elective (we limited our class to 50 and then to 100 students as we fine-tuned the course, and there had been a waiting list), and later as an alternative for computer science and engineering students to the mandatory Mathematical Logic 101, to the clear satisfaction of the students, who continuously rank this course highly. In our experience, having the tasks of a single chapter due each week (if the schedule permits, then we try to allow an additional week for Chapter 10), with the tasks of the first part of this book (Chapters 1 through 6) being solved by each student individually and the tasks of the second part of this book (Chapter 7 through 12) being solved in pairs, has consistently proven to work well.

We are grateful to the Hebrew University students who took our course for their valuable questions and comments, and to our earlier students also for the faith they have put in us. We our indebted to our first TA and beta-solver Alon Ziv, as well as to our subsequent TAs Noam Wies, Asaf Yehudai, Ofer Ravid, and Elazar Cohen, and beta-solvers Omri Cohen and Matan Harsat. A special thanks goes to Chagit Schiff-Blass, at the time a Law and Cognitive Science student, who showed us that our way of teaching Mathematical Logic really does appeal to an even more diverse student population than we had imagined, by first being an excellent beta-solver and then joining our teaching team. We thank Henry Cohn for valuable advice, and thank Aviv Keren and Shimon Schocken for their valuable detailed feedback on portions of earlier drafts of this book. We especially thank David Kashtan for careful and valuable scientific editing of this book on the logic side; any deviations from standard definitions or nomenclature are, of course, our own responsibility. The cover picture by Vasily Kandinsky is titled "Serious-Fun," and we hope that this will describe your experience as you work through this book. We always appreciate feedback from readers.

Yannai A. Gonczarowski
Noam Nisan