

From Hilbert's Program to a Logic Tool Box

Version 1.0

Johann A. Makowsky

Department of Computer Science
Technion – Israel Institute of Technology
Haifa, Israel

janos@cs.technion.ac.il
www.cs.technion.ac.il/~janos

My own research background:

1970-85: Mathematical Logic, Classical Model Theory,
Abstract Model Theory and Generalized Quantifiers;

1980-95: Application of Logic to Semantics, Logic Programming and Databases;

1995-today: Application of Logic to Algorithmics and Combinatorics.

My own undergraduate teaching experience:

- Logic for Computer Science,
Sets and Logic for Computer Science
- Database Systems, Database Theory
- Foundations of Logic Programming,
Foundations of Automated Theorem Proving

What to teach from logic?

In this lecture I want to examine what we should teach from logic to our

non-specialized undergraduate students.

I mean, what does **every** graduate of Computer Science have to learn in/from logic?

The current syllabus is often justified

more by **the traditional narrative**

than by **the practitioner's needs.**

Outline of the talk

Part 1: The Logical Foundations of Mathematics

- From Frege to Gödel: the traditional narrative
- Sets for the Working Mathematician: the practical narrative

Part 2: Lessons for the Working Computer Science Graduate

- Sets as universal data structure
- Computability
- Syntax and semantics
- Definability and interpretability
- My Tool Box

From Frege to Gödel:
The traditional narrative

Logical Foundations of Mathematics

Act I: Cantors Paradise

- First G. Cantor (1874 - 1884) created the Paradise of Sets
- Then G. Frege (1879) created the modern Logical Formalisms, including the **correct binding rules for quantification**, and
- set out to lay the Foundations of Mathematics with his Die Grundgesetze der Arithmetik, Volume1 (1893).
- G. Peano, author of "The principles of arithmetic, presented by a new method" (1889), wrote a positive review of it.

Frege's Program:

Explain all of Mathematics
using Logic (and Set Theory).

Logical Foundations of Mathematics

Act II: Paradise lost

- On 16 June 1902, Bertrand Russell pointed out, with great modesty, that the Russell paradox gave a **contradiction** in Frege's system of axioms.
- And with Russell's paradox started the **crisis** of the **Foundations of Mathematics**,
- G. Cantor had sensed this, when he noticed trouble with the "set of all sets".

Let V be the set of all sets. Then its power set $P(V) \subset V$.
But $|V| < |P(V)|$, a contradiction.

Logical Foundations of Mathematics

Act III: Hilbert's Program

(quoted from Wikipedia)

D. Hilbert around 1920 designs a program to provide secure foundations for all mathematics. In particular this should include:

- **Formalization** of all mathematics: all mathematical statements should be written in a precise formal language, and manipulated according to well defined rules.
- **Completeness**: a proof that all true mathematical statements can be proved in the formalism.
- **Consistency**: a proof that no contradiction can be obtained in the formalism of mathematics. This consistency proof should preferably use only "finitistic" reasoning about finite mathematical objects.

Logical Foundations of Mathematics Hilbert's Program (contd)

- **Conservation:** a proof that any result about "real objects" obtained using reasoning about "ideal objects" (such as uncountable sets) can be proved without using ideal objects.
- **Decidability:** there should be an algorithm for deciding the truth or falsity of any mathematical statement.

In 1928, D. Hilbert and W. Ackermann publish "Grundzüge der theoretischen Logik".

- The Logic in question is **Second Order Logic**.
- What we call **First Order Logic**, is called there the **restricted calculus**.
- They **prove** soundness of the calculus, and ask the question of **completeness**.

Logical Foundations of Mathematics

Act IV: Rise and Fall of Hilberts Program

Initial successes:

- Leopold Löwenheim (1915), Thoralf Skolem (1920), Mojżesz Pressburger (1929), Alfred Tarski (1930), Frank Plumpton Ramsey (1930), László Kalmár (1939) and many others prove partial **decidability** results for fragments of Logic, and for Arithmetic, Algebra, Geometry.
- In 1929 Kurt Gödel proves the **completeness** of the Hilbert-Ackermann axiomatization of the the **restricted** (first order) calculus.

Logical Foundations of Mathematics Rise and Fall of Hilberts Program (contd)

Final blows:

- 1931 K. Gödel proves that every recursive theory which contains arithmetic is **incomplete**.
- 1931 K. Gödel proves that every recursive consistent theory which contains arithmetic cannot prove its own consistency.
- 1936 Alonzo Church and Alain Turing show that already for the **restricted** calculus with free relation variables the set of tautologies is not computable (but is semi-computable).

Logical Foundations of Mathematics

Act V: Clarifications and repairs

Proof Theory arises from work by W. Ackermann, G. Gentzen, J. Herbrand, D. Hilbert and P. Bernays.

Set Theory arises from work by E. Zermelo, D. Mirimanoff, J. von Neumann, A. Fränkel, K. Gödel and P. Bernays.

Alternative approaches are developed by, among others, W. Quine, W. Ackermann, and J.L. Kelley and A.P. Morse

Recursion Theory arises from work by E. Post, J. Herbrand, K. Gödel, A. Church, A. Turing, H. Curry.

Model theory arises from work by T. Skolem, A. Tarski, A. Robinson, R. Fraïssé and A. Mal'cev,

And for long this remained the classical divide of **Mathematical Logic**.

Interlude: The Foundations of Modern Analysis

A **pragmatic** Frege program

It used to be customary to teach the foundations of calculus by:

- Starting with sets.
- Defining the number systems \mathbb{N} , \mathbb{Z} , \mathbb{Q} and their arithmetic operations **inductively** and using **quotient structures**.
- Defining the reals \mathbb{R} , using Dedekind cuts.
- Defining structures, say, groups, fields, Banach spaces, Lie algebras, **axiomatically**.
- **Existence** of axiomatically defined objects had to be established by an **explicit sequence of set construction steps within the cumulative hierarchy**.

Logical Foundations of Mathematics Act VI: 100 years later - Fixing Frege

C. Wright, P. Geach and H. Hodes suggested, and G. Boolos proved (1987) that the modified Frege program actually is feasible.

G. Boolos, On the Consistency of Frege's Foundation of Mathematics, reprinted in: G. Boolos, Logic, Logic and Logic, Harvard University Press, 1998.

So we have:

Frege: The Peano Postulates can be deduced in dyadic second order logic from Hume's principle and suitable definitions of the natural numbers (Frege's Arithmetic).

Boolos: Frege's arithmetic is interpretable in second order Peano Arithmetic.

J.P. Burgess, Fixing Frege, Princeton University Press, 2005

That much for the "big crisis".

The classical textbooks in Logic

The Classical Texts follow this narrative:

- Logic is needed to resolve the paradoxes of set theory.
- First Order Logic is THE LOGIC due to its completeness theorem.
- The main theorems of logic are the **Completeness Theorem** and the **Compactness Theorem**
- The tautologies of First Order Logic are not recursive.
- Arithmetic Truth is not recursively enumerable.
- One cannot prove CONSISTENCY within rich enough systems.

This is NOT what a
Practitioner of Computing Sciences
NEEDS !

So WHAT does a
Practitioner of Computing Sciences
NEED ?

Knowledge of *theoretical orientation*
VS
practical knowledge

Theoretical orientation:

- **awareness** that his domain of discourse is an **idealized world of artefacts** which models fairly accurately the artefacts which allow us to run and interact with computing machinery.
- **awareness** of the different levels of abstractions.
- **awareness** that in this world of artefacts there are **a priori limitations**. Not everything is realizable, computable, etc.

Practical knowledge:

- **tools** which allow him to **model new artefacts**, whenever they arise;
- **tools** which allow him to **prove properties** of the modeled artefacts.

He needs a carefully adapted blend of

- the practical Frege program, with
- the knowledge of its limitations.

He needs both **proficiency** and **performance** in his practical knowledge.

Lessons from 150 years of Modern Logic
and the Foundations of Mathematics

1. Modeling the world

Our scientific language: Natural Language enhanced by precise use of boolean operations, quantification and the use of naive language of sets.

Our universal data structure: A cumulative world of sets.

Modeling the world: We model ALL artefacts of our computing world by constructed objects in the world of sets.

Modeling involves side effects:

Modeled artefact have properties not intended.

Ordered pairs:

N. Wiener: $(x, y) := \{\{\{x\}, \emptyset\}, \{\{y\}\}\}$, K. Kuratowski $\{\{x\}, \{x, y\}\}$,

Simplified $\{a, \{a, b\}\}$.

Fixing levels of abstraction: Introducing structures, and fixing which sets are not further to be analyzed.

A graph is a pair $\langle V, E \rangle$. A finite automaton is a tuple $\langle S, \Sigma, R, I, T \rangle$.

Lesson 1 (contd)

Like in the foundations of Analysis, as practiced by R. Dedekind, E. Landau and N. Bourbaki, we need the **precise language mix** of **normalized natural language** augmented by the **language of sets** to **model** the **idealized artefacts** of computer science.

Artefacts:

strings, concatenation, natural numbers, graphs, relational structures stacks, arrays; circuits, Turing machines, register machines; specification and programming languages,

Tools: Inductive definitions, proofs by induction; enumerations, proving countability and uncountability; well-orderings (for termination)

Is this not "too denotational" ?

... our friends may ask.

- It **does** map everyting into **sets**.
- But "truth" does not **presuppose** a world of sets.
- Truth in the sense of Frege's world is defined by the laws (introduction and elimination rules) of logic and of the Fregean constructs.
- It leaves your **foundational options** open ...

2. Modeling Computability and its limitations (when modeled)

Computability is modeled over different domains, computing operations, resource restrictions.

Natural numbers and recursion:

The original definition of the set of **recursive functions**.

Natural numbers and register machines:

Close to early programming languages.

Turing machines and words: Close to assembly languages.

Other models: Logic programs, Lambda calculus, cellular automata, quantum computing

Showing their equivalence involves

- Translation between the domains.
- Translations between programs (interpreters and compilers).

Lesson 2 (contnd)

Computability may be taught before logic in a more naive way.

Here I want to stress The different basic structures involved, and their bi-interpretability.

Orientation:

Not everything is computable.

Precise statement of various versions of the the Church Turing Hypothesis.

Separating slogans from precise definitions

Effectively computable = P, NP, RP, QP, ?

Tools:

Proving non-computability.

Establishing simulations.

3. Modeling Syntax and Semantics

We look at Propositional, First Order, Second Order Logic, or any other logic of **assertions**.

Syntax:

The syntax is an inductively defined set of words, the well formed expressions.

Semantics:

Structures are interpretations of the basic non-logical symbols.

Assignments are interpretations of the variables.

The **meaning function** associates with structures, assignments, and formulas a truth value.

What is the meaning of an assertion ?

Lesson 3 (contd)

The meaning of an assertion is:

Without free variables: A truth value.

But this is misleading!

With free variables: The set of interpretations of its free variables.

Only first order variables: A relation

We define usually **logical validity** via truth values.

It would be preferable to define

validity and **logical consequence** directly

for formulas **with free variables**.

Do we need the Completeness Theorem?

For the practical knowledge we need:

- The semantic notion of **logical consequence**.
- Enough basic logical equivalences to to prove the **Prenex Normal Form Theorem (PNF)**.
- Introduction and elimination rules for quantifiers (via constants).
- A **game theoretic** interpretation of formulas in PNF.

For the knowledge of orientation we might state (but not prove) the Completeness Theorem for our redundant set of manipulation rules.

Arguments for/ against proving the Completeness Theorem

The classical argument pro:

- Completeness and its corollary, **Compactness** is at the heart of logic.

My arguments contra:

- None of these are part of the practical knowledge we aim at.
- The proof of the Completeness Theorem is a waste of time at the cost of teaching more important skill of understanding the manipulation and meaning of formulas.
- First Order Logic is not privileged in our context. We deal very often with finite structures, where the Completeness Theorem is not true.
- **Second Order Logic** anyhow is the natural logic we work in, and not taking that seriously confuses the student.

Lesson 3 (contd)

We should concentrate on understanding quantification:

Tools:

- Read, write and **understand the meaning** of First Order and **Second Order** formulas.
- Understand the relationship between **projection of relations** and quantification.
- Understand that **Relational Calculus** and First Order Logic are really the same (i.e., bi-interpretable).
- Play with the game interpretation of quantifiers to analyze the **amount of quantification** needed to express, say "there exists at least n elements x such that $\phi(x)$ ".

4. Limitations of formalisms: Definability

Before we prove the Completeness Theorem I would like the students to understand the difference between First Order (FO) and Second Order (SO) Logic.

- "There are an equal number of x with $P(x)$ and $Q(x)$ "
where P, Q are unary predicate symbols, is expressible in SO but not in FO.

We can prove this having the games available.

- In the natural numbers \mathbb{N} , **multiplication** is SO-definable using addition only, but not FO-definable.

Multiplications is FO-definable using addition and squaring.

The negative result we can not prove in an undergraduate course, as we need the decidability of FO Presburger Arithmetic. But we can explain it.

5. Intepretability and Reducibility

Before we prove the Completeness Theorem I would like the students to understand what it means that

The integers \mathfrak{Z} with their arithmetic are **interpretable** inside the natural numbers \mathfrak{N} with their arithmetic.

We define a **transduction** $T(\mathfrak{N}) = \mathfrak{Z}$ as follows.

- The new universe consists of equivalence classes of pairs of natural numbers such that $(x, y) \sim (x', y')$ iff $x + y' = x' + y$.
- The new equality is this equivalence.
- The new addition is the old addition on representatives.
- Same for multiplication.

Lesson 5 (contd)

We define the **intepretation** $S : Formulas \rightarrow Formulas$ as follows:

For any SO-formula ϕ we let $S(\phi)$ be the result of substituting the **new** definitions of addition and multiplication and equality for the corresponding symbols.

S and T are intimately related:

$$\exists = T(\mathfrak{N}) \models \phi \text{ iff } \mathfrak{N} \models S(\phi)$$

which is the **Fundamental Property** of **Transductions** and **Interpretations**.

Lesson 5 (contd)

In the same way we can see that

- The cartesian product is interpretable in the disjoint union.
- Many graph transformations are given as transductions.
- All implementations of one data structure in another are of this form.
- Transductions and interpretations are everywhere

The Fundamental Properties of SO

- Isomorphic structures satisfy the same SO sentences
- That Fundamental Property of Transductions and Interpretations.
- The Prenex Normal Form Theorem and its visualization as a two person game.

The Fundamental Properties of FO

Besides the properties of SO we have

The **Ehrenfeucht-Fraïssé Theorem**:

Two structures **can be distinguished** by a sentences of quantifier depth k iff Player I can force a win in the EF-game of length k .

or, equivalently

Two structures **cannot be distinguished** by a sentences of quantifier depth k iff they are k -isomorphic.

Furthermore:

k -isomorphism is preserved under the formation of disjoint unions of structures.

Modified versions also hold for Monadic Second Order Logic, but **not** for SO.

Combining EF-Games and Transductions

Combining games and transductions gives a very powerful tool to compute the meaning function of a FO formula in a complex structure by reducing this computation to simpler structures.

If G is obtained from graphs H_1, H_2 by applying disjoint unions, cartesian products, and first order definable transductions T_1, T_2 , say

$$G = T_1(H_1 \times T_2(H_2))$$

then the truth of the formulas of quantifier rank k in G is uniquely and effectively determined by the the truth of the formulas of quantifier rank k which hold in H_1 and H_2 .

This is the **Feferman-Vaught Theorem**.

My Logic Tool Box

- The Fundamental Property of Transductions and Interpretations.
- The **Ehrenfeucht-Fraïssé Theorem** and its refinements.
- The Feferman-Vaught Theorem and its variations.

Note that all these tools were already developed before 1960 and widely used by specialists, but not included in textbooks.

I have surveyed how to use these tools in Computer Science in my paper

Algorithmic uses of the Feferman-Vaught theorem,
Annals of Pure and Applied Logic, vol. 126 (2004) pp. 159-213

Where these tools work

Linear Algebra in Math is pervasive.

It is important that the student sees the tools work in the basic courses.

They are (or could be) at work in

- Automata Theory
- Database Systems
- Graph Algorithms
- Decidable FO theories

I have not discussed Automated Theorem Proving where, indeed, Deduction Systems are central.

But in Automated Theorem Proving in Geometry these tools are also essential.

THANK YOU FOR YOUR ATTENTION !

Especially after the wild banquet and dancing of the past night