Logic as The Calculus of Computer Science

Phokion G. Kolaitis

University of California, Santa Cruz

Moshe Y. Vardi

Rice University

Background

E.P. Wigner (1960): *On the Unreasonable Effectiveness of Mathematics in the Natural Sciences*

- "The book of nature is written in the language of mathematics" (attributed to Galileo).
- Numerous examples of the effectiveness of mathematics in the physical sciences
- *The empirical law of epistemology:* The mathematical formulation of the laws of nature is both appropriate and accurate; mathematics is the *correct* language for formulating the laws of nature.

Question: What is the correct language for computer science?

Mathematical Logic

Mathematical Logic was developed in an attempt to confront the crisis in the foundations of mathematics at the turn of the 20th Century.

Hilbert's Program (1900-1928):

Formalize mathematics and establish that:

- Mathematics is *consistent*: a mathematical statement and its negation cannot ever both be proved.

- Mathematics is *complete*: all "true" mathematical statements can be "proved".

- Mathematics is *decidable*: there is a mechanical rule to determine whether a given mathematical statement is "true" or "false".

Hilbert

Hilbert firmly believed that these goals can be achieved:

> “Every mathematical problem must necessarily be susceptible to an exact statement, either in the form of an actual answer to the question asked, or by the proof of the impossibility of its solution.”
>
> “Once a logical formalism is established one can expect that a systematic, so-to-say computational, treatment of logic formulas is possible, which would somewhat correspond to the theory of equations in algebra.”

The Demise of Hilbert's Program

- K. Gödel (1931-3):
  - *Incompleteness* of ordinary arithmetic
  - *Impossibility* of proving consistency of set theory

- A. Church and A. Turing (1936-1937):

  Undecidability of first-order logic:

  - The set of all valid first-order sentences is *not recursive*.
  - The set of all first-order sentences that are true in arithmetics is *not recursively enumerable*.

Where Is Logic Today?

AltaVista, January 21, 2001:

- Logic and Mathematics: 30,460 hits
- Logic and Philosophy: 43,515 hits
- Logic and Computer Science: 44,331 hits

Logic in Computer Science

- During the past 30 years, there has been an extensive and growing interaction between logic and computer science.

- Concepts and methods of logic occupy a central place in computer science, insomuch that logic has been called *the calculus of computer science*.

- Logic has been much more effective in computer science than it has been in mathematics.

M. Davis (1988): *Influences of Mathematical Logic on Computer Science*:

> "When I was a student, even the topologists regarded mathematical logicians as living in outer space. Today the connections between logic and computers are a matter of engineering practice at every level of computer organization."

Why?

Alan Turing:

> “I expect that digital computing machines will eventually stimulate a considerable interest in symbolic logic ... The language in which one communicates with these machines .. forms a sort of symbolic logic.”

Logic provides:

- formalisms to describe mathematical structures
- languages to describe properties of mathematical structures
- languages to describe dynamic processes
- a clear distinction between syntax and semantics

Michael Jackson: “Description is our business”

Computer Science Started as Logic

In the beginning, there was logic:

- Outcome of Hilbert's program: formalization of algorithms (Church's Thesis)
- Stored-progarm computer: universal Turing machine
- Functional programming: λ-calculus
- Recursive-function theory: precursor of complexity theory
- Automata theory: finite-state automata

That was then, this is now!

Logic and Computer Science

The goal of this workshop is to provide evidence in support of the unusual effectiveness of logic in computer science by highlighting a few of the areas of computer science in which logic has had a definite and lasting impact.

Formalisms

Several different logical formalisms are involved in the interaction between logic and computer science.

We will briefly review the basics of three of them:

- Propositional Logic (a.k.a. Boolean Logic)
- First-Order Logic
- Modal Logic

Propositional Logic

A formalism for modeling statements that are either *true* or *false*

Syntax:

- Propositional Variables: P, Q, R, . . .

- Propositional Formulas:
  variables $+$ connectives ($\neg$, $\vee$, $\wedge$, $\rightarrow$, . . .)

  Examples:

$$P \vee \neg Q$$
$$(P \wedge (P \rightarrow Q)) \rightarrow Q$$

Semantics:

- Truth assignment $\nu : \{P, Q, R, \ldots\} \mapsto \{0, 1\}$

- Extend to propositional formulas $\nu(\psi)$ using truth tables.

  Examples: Given $\nu(P) = 0$ and $\nu(Q) = 1$,

$$\nu(P \vee \neg Q) \;=\; 0$$

$$\nu((P \wedge (P \rightarrow Q)) \rightarrow Q) = 1$$

Algorithmic Problems in Propositional Logic

Truth Evaluation Problem (Model Checking):

Given a propositional formula ψ and a truth assignment ν, what is the value of $\nu(\psi)$?

Satisfiability Problem:

Given a propositional formula ψ, is there a truth assignment ν such that $\nu(\psi) = 1$?

Examples:

- $(P \vee Q) \wedge (\neg P \vee Q) \wedge (P \vee \neg Q)$ is satisfied by $\nu(P) = 1$, $\nu(Q) = 1$.
- $(P \vee Q) \wedge (\neg P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee \neg Q)$ is *unsatisfiable*.

Remarks:

- Truth evaluation is in PTIME.
- The Satisfiability Problem is *NP-complete*.

Before Cook

Inference Problem:

Given propositional formulas φ and ψ, is ψ a consequence of φ?

This problem is *co-NP-complete*.

Ernst Schröder, 1890-1910: "The Algebra of Logic"

> "getting a handle on the consequences of any premisses, or at least the fastest method for obtaining these consequences, seems to me to be one of the noblest, if not the ultimate goal of mathematics and logic."

First-Order Logic

A formalism for specifying properties of mathematical structures, such as *graphs*, *partial orders*, *groups*, *rings*, *fields*, . . .

For simplicity, we focus on *Relational Structures*

$$\mathbf{A} = (A, R_1, \ldots, R_k),$$

- A is a non-empty set;
- R_i is an m-ary *relation* on A, for some m (that is, R_i is a set of m-tuples from A).

Example:

- Graph $\mathbf{G} = (V, E)$, where $E \subseteq V^2$

Remark:

- A relational structure is essentially a *relational database*.

First-Order Logic on Graphs

Syntax:

- First-Order Variables: x, y, z, . . .
- Atomic Formulas: $E(x, y)$, $x = y$
- Formulas: Atomic Formulas + Connectives + First-Order Quantifiers $\exists x$, $\forall x$, . . .

Examples:

- "node x has at least two distinct neighbors"

$$(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$$

Concept: x is *free* in the above formula

- "each node has at least two distinct neighbors"

$$(\forall x)(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$$

Concept: The above is a *sentence*, that is, a formula with no free variables.

First-Order Logic on Graphs

Semantics:

- First-order variables and quantifiers range over elements of the universes of structures
- To evaluate a formula φ, we need a graph $\mathbf{G}$ and an assignment ν that maps the free variables of φ to nodes of $\mathbf{G}$

  Notation: $\mathbf{G} \models \varphi(x_1/a_1, \ldots, x_k/a_k)$
- Formulas $\varphi(x_1, \ldots, x_k)$ define *queries*:

  $Q_\varphi(\mathbf{G}) = \{(a_1, \ldots, a_k) : \mathbf{G} \models \varphi(x_1/a_1, \ldots, x_k/a_k)\}$
- A sentence ψ is either true or false on a given graph $\mathbf{G}$. In particular, sentences define *Boolean queries*, that is, they specify classes of graphs.

$$Q_\psi = \{\mathbf{G} : \mathbf{G} \models \psi\}$$

Algorithmic Problems in First-Order Logic

Truth Evaluation Problem (Model Checking): Given a first-order formula $\varphi(x_1, \ldots, x_k)$, a graph $\mathbf{G}$, and nodes $a_1, \ldots, a_k$, does $\mathbf{G} \models \varphi(x_1/a_1, \ldots, x_k/a_k)$?

Satisfiability Problem: Given a first-order sentence ψ, is there a graph $\mathbf{G}$ such that $\mathbf{G} \models \psi$?

Remarks:

- Truth evaluation, which is query evaluation, is decidable.
- The Satisfiability Problem is *undecidable*.

Modal Logic

A formalism for modeling *necessity* and *possibility*

Syntax:

Propositional Logic $+$ Necessity Operator $\Box$

Example: $(\Box P \wedge \Box(P \rightarrow Q)) \rightarrow \Box Q$

Possible World Semantics:

- Kripke structure $\mathbf{M} = (S, R, \pi)$:
  - S is the set of possible worlds and $R \subseteq S^2$ is the *accessibility* relation between worlds;
  - $\pi(s)$ is a set of propositional variables, for every world $s \in S$.
- $(\mathbf{M}, s) \models P$ if and only if $P \in \pi(s)$.
- $(\mathbf{M}, s) \models \Box\psi$ if and only if $(\mathbf{M}, t) \models \psi$, for every world t such that $(s, t) \in R$.

Modal Logic

- $\Box\varphi$ may mean “φ *is known*” (epistemic logic), “φ *is always true*” (temporal logic), or “φ *holds after the program is executed*” (dynamic logic).

- Modal logic can be translated into (a small fragment of) first-order logic.

- Modal logic has good algorithmic properties; in particular, the satisfiability problem for modal logic is decidable.

Impact of Logic in CS

Logic has been effective in several different areas of computer science, including

- Computer-Aided Verification (K. McMillan)
- Database Systems (V. Vianu)
- Computational Complexity (N. Immerman)
- Programming Languages (P. Lee)
- Computer Security (J.K. Millen)

as well as

- Artificial Intelligence
- Digital Design
- Distributed Systems
- Logic Programming
- Software Engineering
- . . .

Computer-Aided Verification

Key Idea of Model Checking:

- Model design (HW or SW) as a finite Kripke structure M
- Express specification as a temporal logic formula ψ
- Check that $M \models \psi$
- Use special data structures to handle very large state spaces
- A negative answer is accompanied by a counterexample

Impact:

- Technology is spreading in the HW industry
- Initial penetration of SW industry

Logic and Databases

A multibillion-dollar success:

- First-order logic lies at the core of modern database systems (SQL,QBE)

- Efficient query evaluation is based on *relational algebra* (a.k.a. cylindrical algebra)

Current challenge: Semistructured data

- *XML*: trees

- *Data Type Definition (DTD)*: tree automata

- *XML queries*: tree transducers

Logic and Complexity

Complexity:

- *Computational Complexity*: amount of resources, such as time or space, required by a machine that solves a problems
- *Descriptive Complexity*: syntatic complexity of describing a problem in some logical formalism

An amazing connection: computational complexity and description complexity are intimately related.

$$\begin{array}{rcl} \mathrm{FO(TC)} & = & \mathrm{NLOGSPACE} \\ \mathrm{FO(LFP)} & = & \textsf{PTIME} \\ \mathrm{SO}\exists & = & \textsf{NP} \\ \mathrm{FO(PFP)} \;=\; \mathrm{SO(TC)} & = & \mathrm{PSPACE} \end{array}$$

Remark:

- right column: machine based
- left column: machine independent

Logic and Programming Languages

Coherence: From *Webster's Collegiate Dictionary*:

- The quality or state of cohering: as
  - systematic or logical connection or consistency
  - integration of diverse elements, relationships, or values

Research in foundations of programming languages strives to bring coherence in software systems, in both senses. Logic has been the primary tool:

- formal semantics
- type theory
- rewriting systems
- . . .

Logic is not only used to design languages and compilers, but is also applied directly to real software artifacts.

Logic and Computer Security

Pervasiveness and diversity

Areas of applicability: numerous!

- Multilevel OS security
- Access control policies
- Public-key infrastructure and trust management
- Cryptographic-protocol analysis

Means of application: numerous!

- Authentication logics
- Temporal logics
- Linear logics
- Model checking
- Theorem proving
- Logic programming

Logic from Computer Science

Computer science problems invigorate logic research:

- Unlike traditional mathematical logic, primary focus on the study of logics on finite structures (e.g., databases): *finite-model theory*.
- Powerful extensions of first-order logic are considered: e.g., *fixpoint logics* in databases.
- Emphasis on connections between logic and automata, especially automata on infinite objects: e.g., *Rabin tree automata* in verification.
- Major role for modal logics: e.g., *description logics* in AI.
- New logics have to be invented: e.g., authentication logics in security
- New notions of inference: e.g., *non-monotonic inference* in AI.

Logic in Computer Science

- Logic is mainly used as a formalism for describing objects and specifying their properties.

- Logic in Computer Science is an *applied science*, combining foundational research with applications.