Logic in Computer Science: An Algorithmic Perspective

Moshe Y. Vardi

http://www.cs.rice.edu/~vardi/comp409

February 1998

COMP 409: Senior-Level Logic in CS

Focus: Study logic as a formal language

- Propositional, 1st-order
- Syntax
- Semantics
- Validity and satisfiability
- Expressibility
- Provability

No: modal logic, temporal logic, etc.

Computer Science Perspective:

algorithms and complexity

Syntax

The Compiler's Perspective:

- Syntax via Backus-Naur Form (BNF)
- Wffs as a context-free language
- Unique readability as unambiguous parsing
  - Parentheses vs. precedence
- Succinctness: Formulas vs. circuits (trees vs. dags)
- Algorithms to recognize wffs

Complexity-Theoretic Background

$$LOGSPACE \subseteq P \subseteq NP \subseteq PSPACE$$

Known:

- $LOGSPACE \subset PSPACE$: proper containment

Unknown: proper containments

- $LOGSPACE \subset P$
- $P \subset NP$
- $NP \subset PSPACE$

Mystery: Where does the chain break?

Propositional Satisfiability and Validity

- *Motivation*: Hardware verification ("Verity – A formal verification program for custom CMOS circuits")
- *Complexity*: P vs. NP vs. co-NP
  - *Satisfiability*: NP-complete
  - *Validity*: co-NP-complete
- *Algorithms*: Truth-Table Method vs. Davis-Putnam Procedure
  - DPP is equivalent to TMM in terms of correctness
  - DPP has exponential-time worst-case complexity
  - DPP performs quite well on practice

Propositional Equivalence and Automata

- *Input*: formulas $\varphi(P_1, \ldots, P_n)$, $\psi(P_1, \ldots, P_n)$
- *Truth assignment*: member of $\{0,1\}^n$
- *Model set*: $models(\varphi) \subseteq \{0,1\}^n$
- *Algorithm*:
  - Construct finite-state automata A_φ and A_ψ such that $L(A_\varphi) = models(\varphi)$ and $L(A_\psi) = models(\psi)$
  - check that $L(A_\varphi) = L(A_\psi)$
- *Construction*: Use closure of regular languages under Boolean operations.

Key Idea: canonicity of deterministic-finite state automata.

Fact: method is used in industry for checking equivalence of circuits (via BDDs).

Horn Formulas

Definition:

- *Literal*: p or $\neg p$
- *Clause*: disjunction of literals
- *Horn Clause*: clause with at most one positive literal, e.g., $p_1 \wedge \ldots \wedge p_k \rightarrow q$
- *Horn Formula*: conjunction of Horn clauses
- *Application*: logic programming

Surprise: Davis-Putnam procedure (which is a heuristics for SAT) is a polynomial-time algorithm for HORN-SAT.

Lesson: Heuristics can be guaranteed to work in special cases

Provability

Mathematics vs. Computer Science:

- *Mathematical logic focus*: minimality
  - Axiom:

$$((((p \rightarrow q) \rightarrow (\neg r \rightarrow \neg s)) \rightarrow r) \rightarrow u) \rightarrow \\ ((u \rightarrow p) \rightarrow (s \rightarrow p))$$

  - M.P.: $\frac{p, p \rightarrow q}{q}$
- *Computer science focus*: efficiency
  - De Morgan Laws: convert to CNF
  - Resolution: $\dfrac{\varphi \vee p, \psi \vee \neg p}{\varphi \vee \psi}$

Surprise: Resolution is a polynomial-time algorithm for 2-SAT (conjunction of binary clauses).

First-Order Logic

FOL as a database query languages:

- *Relational Structures*=*Relational Databases*
- *Formulas*=*Queries*
- $\mathcal{A} \models \phi$=Query Evaluation

Crux: focus on *finite* structures

Eval($\mathcal{A}, \varphi$)

- *Algorithm*
- *Complexity in* $|\varphi|$: PSPACE
- *Complexity in* $|\mathcal{A}|$: LOGSPACE

Lesson: complexity in what?

Provability

- Soundness vs. completeness
- Provability vs. (polynomial time) proof checking
  - Tautologies: Bad axioms
  - Resolution: polynomial-time checking (unification)
- Provability vs. automated theorem proving: decision vs. search

Incompleteness and Undecidability

- recursive vs. recursive enumerability vs. non-recursive-enumerability: *computable=r.e.*
- *Surprise*: recursive enumerability of validity
- *Bad News*: validity is not recursive
- *Incompleteness*: recursive non-enumerability (*no* Gödel Incompleteness Theorem)
- Incompleteness is pervasive:
  - Validity and finite satisfiability: r.e. but not recursive
  - Satisfiability and finite validity: not r.e.

Context is Everything

Logic always has a context: semantical domain

- Validity over *all* structures: r.e. but not recursive
- Validity over *all finite* structures: not r.e.
- Validity over *integers*: not r.e.
- Validity over *reals*: decidable