ECE650

Boolean Logic

Vijay Ganesh

What is this Lecture About?

- This lecture is about mathematical logic and theory of computation, and their applications.
- You will learn boolean logic, first-order logic, their properties, as well as theoretical models of computation and complexity
- You will learn about SAT/SMT solvers and their applications in software engineering, security, and AI

What is Logic?

- A precise language, with clear syntax and well-defined semantics, for mathematics and computation
- Study of the foundations of mathematics, i.e., axiomatization and proof systems
- Computation, aimed at automation of mathematics

Why Should You Care About Logic?

Logic is a fundamental part of computer science:

- Computation, irrespective of representation, can be very complex to understand/process in all its gory detail.
- Hence, we need abstractions.
- Logics are precise languages that allow us to represent/manipulate/process/morph abstractions of computations.
- Examples include Boolean logic (aka propostional or sentential calculus), predicate logic, first-order theories,...
- Claude Shannon (inventor of information theory) realized in the 1940's that all digital hardware can be represented in Boolean logic, considerably easing hardware design

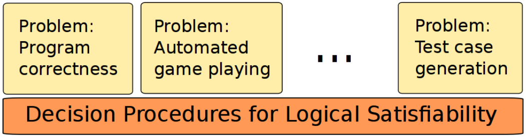
Why Should You Care?

Logic is a fundamental part of computer science:

- Artificial intelligence: constraint satisfaction, automated game playing, planning, . . .
- Programming Languages: logic programming, type systems, programming language theory . . .
- Hardware verification and synthesis: correctness of circuits, ATPG, . . .
- Program analysis, verification and synthesis: Static analysis, software verification, test case generation, program understanding, . . .

Why Should You Care?

- No matter what your research area or interest is, the techniques we cover in this course are likely to be relevant.
- Very good tool kit because many difficult problems can be reduced deciding satisfiabilty of formulas in logic.


Problem: Program correctness
Problem: Automated game playing
...
Problem: Test case generation
Decision Procedures for Logical Satisfiability

Context and Motivation for Logic and Theory of Computation

- Logic started, in many cultures but particularly in ancient Greece, as a way of making language precise and arguments sound (i.e., rules of reasoning that don't lead to contradictions)

- Another impetus for logic was the idea of axiomatization of Geometry by Euclid 2500 years ago (however, it was not recognized as part of logic until much later)

- George Boole publishes *Laws of Thought* in 1854. Key contribution - logic can be given mathematical foundations

- Gottleib Frege publishes *A Formal Language for Pure Thought Modeled on that of Arithmetic* in 1879. Key contribution - Not only can logic be given mathematical foundations, but also that all of mathematics can be derived from logic

Context and Motivation for Logic and Theory of Computation

- George Cantor proves an astounding result that there are multiple levels of infinities in his paper *On a Property of the Collection of All Real Algebraic Numbers* in 1874. He goes on to invent set theory. Key contributions - invents set theory, completely changes mathematics, foundational ideas for computer science

- Bertrand Russell finds a "bug" in Frege's foundations of set theory in 1901 - Russell's paradox

- Peano, Zermelo, and Frankel develop axioms for all of mathematics (ZF and ZFC set theories) in 1905

- In a stunning proof in 1931, Godel shows that all consistent formal axiomatic systems for sufficiently strong arithmetic must be incomplete

- In another stunning set of results in 1936, Turing invents the notion of computation and shows that first-order logic is undecidable

Review of Propositional Logic: Syntax

Atom	truth symbols $\top$ ("true") and $\bot$ ("false") propositional variables $p, q, r, p_1, q_1, r_1, \cdots$
Literal	atom α or its negation $\neg\alpha$
Formula	literal or application of a logical connective to formulae F, F_1, F_2

$\neg F$	"not"	(negation)
$F_1 \wedge F_2$	"and"	(conjunction)
$F_1 \vee F_2$	"or"	(disjunction)
$F_1 \rightarrow F_2$	"implies"	(implication)
$F_1 \leftrightarrow F_2$	"if and only if"	(iff)

Interpretations in Propositional Logic: Semantics

- An interpretation I for a formula F in propositional logic is a mapping from each propositional variables in F to exactly one truth value

$$I : \{p \mapsto \top, q \mapsto \bot, \cdots\}$$

- For a formula F with 2 propositional variables, how many interpretations are there?
- In general, for formula with n propositional variables, how many interpretations?

Entailment: Semantics

- Under an interpretation, every propositional formula evaluates to T or F

  Formula F + Interpretation I = Truth value

- We write $I \models F$ if F evaluates to $\top$ under I (satisfying interpretation)

- Similarly, $I \not\models F$ if F evaluates to $\bot$ under I (falsifying interpretation).

Inductive Definition of Propositional Semantics

Base Cases:

$$I \models \top \qquad I \not\models \bot$$

$$\begin{aligned} I \models p \quad &\text{iff} \quad I[p] = \top \\ I \not\models p \quad &\text{iff} \quad I[p] = \bot \end{aligned}$$

Inductive Cases:

$$\begin{aligned}
I \models \neg F \quad & \text{iff } I \not\models F \\
I \models F_1 \wedge F_2 \quad & \text{iff } I \models F_1 \text{ and } I \models F_2 \\
I \models F_1 \vee F_2 \quad & \text{iff } I \models F_1 \text{ or } I \models F_2 \\
I \models F_1 \rightarrow F_2 \quad & \text{iff, } I \not\models F_1 \text{ or } I \models F_2 \\
I \models F_1 \leftrightarrow F_2 \quad & \text{iff, } I \models F_1 \text{ and } I \models F_2 \\
& \quad \text{or } I \not\models F_1 \text{ and } I \not\models F_2
\end{aligned}$$

Simple Example

- What does the formula

$$F:\ (p \leftrightarrow \neg q) \rightarrow (q \rightarrow \neg r)$$

evaluate to under this interpretation?

$$I = \{p \mapsto \bot,\ q \mapsto \top, r \mapsto \top\}$$

- $I \not\models F$

Satisfiability and Validity

- F is satisfiable iff there exists an interpretation I such that $I \models F$.
- F valid iff for all interpretations I, $I \models F$.
- F is contingent if it is satisfiable but not valid.
- Duality between satisfiability and validity:

$$\boxed{F \text{ is valid iff } \neg F \text{ is unsatisfiable}}$$

- Thus, if we have a procedure for checking satisfiability, this also allows us to decide validity

Deciding Satisfiability and Validity

- Before we talk about practical algorithms for deciding satisfiability, let's review some simple techniques
- Two very simple techniques:
  - Truth table method: essentially a search-based technique (model-theoretic)
  - Semantic argument method: deductive way of deciding satisfiability (proof-theoretic)
- Completely different, but complementary techniques
- In fact, as we'll see later, modern SAT solvers combine both search-based and deductive techniques!

Method 1: Truth Tables

Example $F : (p \land q) \rightarrow (p \lor \neg q)$

p q	$p \land q$	$\neg q$	$p \lor \neg q$	F
0 0	0	1	1	1
0 1	0	0	0	1
1 0	0	1	1	1
1 1	1	0	1	1

Thus F is valid.

Another Example

$F : (p \lor q) \rightarrow (p \land q)$

p q	$p \lor q$	$p \land q$	F	
0 0	0	0	1	$\leftarrow$ satisfying I
0 1	1	0	0	$\leftarrow$ falsifying I
1 0	1	0	0	
1 1	1	1	1	

Thus F is satisfiable, but invalid.

Summary: Truth Tables

- List all interpretations $\Rightarrow$ If all interpretations satisfy formula, then valid. If no interpretation satisfies it, unsatisfiable.
- Completely brute-force, impractical: requires explicitly listing all 2^n interpretations in the worst-case!
- Method does not work for any logic where domain is not finite (e.g., first-order logic)

Method 2: Semantic Argument

- Semantic argument method is essentially a proof by contradiction, and is also applicable for theories with non-finite domain.
- Main idea: Assume F is not valid $\Rightarrow$ there exists some falsifying interpretation I such that $I \not\models F$
- Apply proof rules.
- If we derive a contradiction in every branch of the proof, then F is valid.
- If there exists some branch where we cannot derive a contradiction (after exhaustively applying all proof rules), then F is not valid.
- A proof system is a collection of proof rules (written in very precise language). The properties of proof systems we care about include soundness, completeness, decidability, complexity.
- A proof is a finite graph, whose nodes are proof rules and edges denote formulas

Properties of Proof Systems

- Proof systems, collection of proof rules, take as input formulas and determine whether they are theorem. (Sometimes we choose to treat the terms theorem and valid as synonyms, even though technically theorem is proof-theoretic concept while validity is model-theoretic)
- Soundness: We say a proof system P is sound for logic (or theory) L, if whenever P returns "theorem" for any L-formula F, then F is indeed valid.
- Completeness: We say a proof system P is complete for logic (or theory) L, if for any valid L-formula F, P returns "theorem".
- Termination: We say a proof system P is terminating, if for any L-formula F (valid or otherwise), P returns after finite number of steps.
- Complexity: Later in the course, we will discuss complexity of proof systems

The Proof Rules (I)

- According to semantics of negation, from $I \models \neg F$, we can deduce $I \not\models F$:

$$\frac{I \models \neg F}{I \not\models F}$$

- Similarly, from $I \not\models \neg F$, we can deduce:

$$\frac{I \not\models \neg F}{I \models F}$$

The Proof Rules (II)

- According to semantics of conjunction, from $I \models F \wedge G$, we can deduce:

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array} \leftarrow \text{and}}$$

- Similarly, from $I \not\models F \wedge G$, we can deduce:

$$\frac{I \not\models F \wedge G}{I \not\models F \quad | \quad I \not\models G}$$

- The second deduction results in a branch in the proof, so each case has to be examined separately!

The Proof Rules (III)

- According to semantics of disjunction, from $I \models F \vee G$, we can deduce:

$$\frac{I \models F \vee G}{I \models F \quad | \quad I \models G}$$

- Similarly, from $I \not\models F \vee G$, we can deduce:

$$\frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

The Proof Rules (IV)

- According to semantics of implication:

$$\frac{I \models F \rightarrow G}{I \not\models F \quad | \quad I \models G}$$

- And:

$$\frac{I \not\models F \rightarrow G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}}$$

The Proof Rules (V)

- According to semantics of iff:

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \quad | \quad I \models \neg F \wedge \neg G}$$

- And:

$$\frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \quad | \quad I \models \neg F \wedge G}$$

The Proof Rules (Contradiction)

- Finally, we derive a contradiction, when I both entails F and does not entail F:

$$\begin{array}{ccc} I & \models & F \\ I & \not\models & F \\ \hline I & \models & \bot \end{array}$$

An Example

Prove $F : (p \land q) \rightarrow (p \lor \neg q)$ is valid.

Let's assume that F is not valid and that I is a falsifying interpretation.

1.	I	$\not\models$	$(p \land q) \rightarrow (p \lor \neg q)$	assumption
2.	I	$\models$	$p \land q$	1 and $\rightarrow$
3.	I	$\not\models$	$p \lor \neg q$	1 and $\rightarrow$
4.	I	$\models$	p	2 and $\land$
5.	I	$\models$	q	2 and $\land$
6.	I	$\not\models$	p	3 and $\lor$
7.	I	$\not\models$	$\neg q$	3 and $\lor$
8.	I	$\models$	$\bot$	4 and 6 are contradictory

$\Rightarrow$ Thus F is valid.

Another Example

- Prove that the following formula is valid using semantic argument method:

$$F : \quad ((p \rightarrow q) \land (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

Equivalence

- Formulas F_1 and F_2 are equivalent (written $F_1 \Leftrightarrow F_2$) iff for all interpretations I, $I \models F_1 \leftrightarrow F_2$

$$\boxed{F_1 \Leftrightarrow F_2 \text{ iff } F_1 \leftrightarrow F_2 \text{ is valid}}$$

- Thus, if we have a procedure for checking satisfiability, we can also check equivalence.

Implication

- Formula F_1 implies F_2 (written $F_1 \Rightarrow F_2$) iff for all interpretations I, $I \models F_1 \rightarrow F_2$

$$\boxed{F_1 \Rightarrow F_2 \text{ iff } F_1 \rightarrow F_2 \text{ is valid}}$$

- Thus, if we have a procedure for checking satisfiability, we can also check implication

- Caveat: $F_1 \Leftrightarrow F_2$ and $F_1 \Rightarrow F_2$ are not formulas (they are not part of PL syntax); they are semantic judgments!

Example

- Prove that $F_1 \wedge (\neg F_1 \vee F_2)$ implies F_2 using semantic argument method.

Summary

- Today:

  Review of basic concepts underlying Boolean (propositional) logic

- We covered syntax, semantics, and proof systems for Boolean logic
- We introduced the concept of satisfiability, validity, proofs, and proof systems
- We covered properties of proof systems such as completeness, soundness, termination
- Next lecture:

  Normal forms and algorithms for deciding satisfiability