About Instructor

- **Research areas**: Formal methods, programming languages, software engineering
- **Specifically**: Program verification/synthesis, program analysis, decision procedures
- **E-mail**: isil@cs.utexas.edu
- **Office hours**: Thursdays 3:30-4:30 pm in GDC 5.726

About TA

- **TA**: Arthur Peters
- **Research**: Programming languages, currently working on Orc
- **E-mail**: amp@cs.utexas.edu
- **Office hours**: 1-2 pm on Monday, noon on Thursday at his desk (GDC 5S)

What is this Course About?

- **This course is about computational logic**.
- **Explore various logical theories widely used in computer science**.
- **Learn about decision procedures that allow us to automatically decide satisfiability and validity of logical formulas**.

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**: correctness of circuits, ATPG, . . .
- **Program analysis**: Static analysis, software verification, test case generation, program understanding, . . .
Overview of the Course

- Part I: Propositional logic and SAT solving
  - Normal forms, DPLL
  - Modern SAT solvers
  - Applications and variations
  - Binary Decision Diagrams
  - Quantified Boolean Logic (guest lecture by Marijn Heule)

Overview, cont.

- Part II: First-order theorem proving
  - Semantics of FOL and theoretical properties
  - Basics of first-order theorem proving
  - Decidable fragments of FOL

Overview, cont.

- Part III: First-order theories and SMT solving
  - Semantics of commonly-used first-order theories
  - Decision procedure for theory of equality
  - Decision procedure for LRA (Simplex)
  - Decision procedure for LIA (branch-and-cut)
  - Combining theories, Nelson-Oppen method
  - DPLL(T) and practical SMT solvers

Overview, cont.

- Part IV: Applications
  - Program verification: Hoare logic, loop invariants, verification conditions
  - Logical abduction, Craig interpolation
  - Constraint simplification, program analysis
  - ACL2 theorem prover (guest lecture by Matt Kaufmann)

Logistics

- Class meets every Tuesday, Thursday 2-3:20 pm
- All class material (slides, relevant reading etc.) posted on the course website:

  http://www.cs.utexas.edu/~idillig/cs395t

- No required books, but following textbooks may be useful

Reference #1

- The Calculus of Computation
  by Aaron Bradley and Zohar Manna
Exams

- Exam dates: Feb 24, March 31, May 5 – put these dates on your calendar!
- All exams closed-book, closed-notes, closed-laptop, closed-phone etc.
- Will give practice problems before exams
- Free during final exam week!

Let’s get started!

- Today: Review of basic propositional logic
- Most of you should already know this stuff – meant as quick refresher!

Review of Propositional Logic: PL Syntax

- Atom: truth symbols ⊤ ("true") and ⊥ ("false")
- Propositional variables p, q, r, p₁, q₁, r₁, · · ·
- Literal: atom α or its negation ¬α
- Formula: literal or application of a logical connective to formulae F₁, F₂
  - ¬F (negation)
  - F₁ ∧ F₂ (conjunction)
  - F₁ ∨ F₂ (disjunction)
  - F₁ → F₂ (implication)
  - F₁ ↔ F₂ (iff)

Interpretations and Entailment

- An interpretation I for a formula F in propositional logic is a mapping from each propositional variables in F to exactly one truth value
  \[ I : \{ p \mapsto \top, q \mapsto \bot, \cdots \} \]
- Under an interpretation, every propositional formula evaluates to \( \top \) or \( \bot \)
- 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)
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
- Semantic argument method: deductive way of deciding satisfiability
- Modern SAT solvers combine search and deduction!
Another Example

\[ F : (p \lor q) \rightarrow (p \land q) \]

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p \lor q</th>
<th>p \land q</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus \( F \) is satisfiable, but invalid.

Bad Idea!

- Truth tables are completely brute-force, impractical \( \Rightarrow \) must list all \( 2^n \) interpretations!
- Does not work for any other 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 is some branch where we cannot derive \( \bot \) (after applying all proof rules), then \( F \) is not valid.

The Proof Rules (I)

- According to semantics of negation, from \( I \models \neg F \), we can deduce \( I \not\models F \):
  \[
  \begin{align*}
  I &\models \neg F \\
  I \not\models F
  \end{align*}
  \]
- Similarly, from \( I \not\models \neg F \), we can deduce:
  \[
  \begin{align*}
  I \not\models \neg F \\
  I \models F
  \end{align*}
  \]

The Proof Rules (II)

- According to semantics of conjunction, from \( I \models F \land G \), we can deduce:
  \[
  \begin{align*}
  I &\models F \land G \\
  I \models F \\
  I \models G \quad \text{← and}
  \end{align*}
  \]
- Similarly, from \( I \not\models F \land G \), we can deduce:
  \[
  \begin{align*}
  I \not\models F \land G \\
  I \not\models F \\
  I \not\models G
  \end{align*}
  \]
- 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 \( I \models F \lor G \), we can deduce:
  \[
  \begin{align*}
  I &\models F \lor G \\
  I \models F \\
  I \models G
  \end{align*}
  \]
- Similarly, from \( I \not\models F \lor G \), we can deduce:
  \[
  \begin{align*}
  I \not\models F \lor G \\
  I \not\models F \\
  I \not\models G
  \end{align*}
  \]
The Proof Rules (IV)

- According to semantics of implication:

\[ I \models F \rightarrow G \]
\[ I \not\models F \rightarrow I \not\models G \]

- And:

\[ I \not\models F \rightarrow G \]
\[ I \models F \]
\[ I \not\models G \]

The Proof Rules (V)

- According to semantics of iff:

\[ I \models F \leftrightarrow G \]
\[ I \models F \land G \]
\[ I \models \neg F \land \neg G \]

- And:

\[ I \not\models F \leftrightarrow G \]
\[ I \models F \land \neg G \]
\[ I \models \neg F \land G \]

The Proof Rules (Contradiction)

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

\[ I \models F \]
\[ I \not\models F \]
\[ I \models \bot \]

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. \( I \not\models \bot \) 4 and 6 are contradictory

\( \Rightarrow \) Thus \( F \) is valid.

Another Example

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

\[ F : ((p \rightarrow q) \land (q \rightarrow r)) \rightarrow (p \rightarrow r) \]

Equivalence

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

\[ F_1 \leftrightarrow F_2 \text{ iff } F_1 \leftrightarrow F_2 \text{ is valid} \]

- 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 $I$, $I \models F_1 \rightarrow F_2$

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

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

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

Example

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

Summary

- Today:
  Review of basic concepts underlying propositional logic

- Next lecture:
  Normal forms and algorithms for deciding satisfiability

- Reading:
  Bradley & Manna textbook until Section 1.6