Course Introduction

Emina Torlak
emina@cs.washington.edu
Today

What is this course about?

Course logistics

Review of basic concepts
Tools for building better software, more easily
Tools for building better software, more easily

more reliable, faster, more energy efficient
Tools for building better software, more easily

automatic verification, debugging & synthesis
Tools for building better software, more easily

class List {
    Node head;

    void reverse() {
        Node near = head;
        Node mid = near.next;
        Node far = mid.next;

        near.next = far;
        while (far != null) {
            mid.next = near;
            near = mid;
            mid = far;
            far = far.next;
        }

        mid.next = near;
        head = mid;
    }
}

class Node {
    Node next; String data;
}
class List {
    Node head;

    void reverse() {
        Node near = head;
        Node mid = near.next;
        Node far = mid.next;

        near.next = far;
        while (far != null) {
            mid.next = near;
            near = mid;
            mid = far;
            far = far.next;
        }
        mid.next = near;
        head = mid;
    }
}

class Node {
    Node next; String data;
}
Tools for building better software, more easily

class List {
    Node head;

    void reverse() {
        Node near = head;
        Node mid = near.next;
        Node far = mid.next;

        near.next = far;
        while (far != null) {
            mid.next = near;
            near = mid;
            mid = far;
            far = far.next;
        }

        mid.next = near;
        head = mid;
    }
}

class Node {
    Node next; String data;
}
Tools for building better software, more easily

class List {
    Node head;

    void reverse() {
        Node near = head;
        Node mid = near.next;
        Node far = mid.next;

        near.next = far;
        while (far != null) {
            mid.next = near;
            near = mid;
            mid = far;
            far = far.next;
        }

        mid.next = near;
        head = mid;
    }
}

class Node {
    Node next; String data;
}
class List {
    Node head;

    void reverse() {
        Node near = head;
        Node mid = near.next;
        Node far = mid.next;

        near.next = ???;
        while (far != null) {
            mid.next = near;
            near = mid;
            mid = far;
            far = far.next;
        }

        mid.next = near;
        head = mid;
    }
}

class Node {
    Node next; String data;
}
class List {
    Node head;

    void reverse() {
        Node near = head;
        Node mid = near.next;
        Node far = mid.next;
        near.next = null;
        while (far != null) {
            mid.next = near;
            near = mid;
            mid = far;
            far = far.next;
        }
        mid.next = near;
        head = mid;
    }
}

class Node {
    Node next; String data;
}
By the end of this course, you’ll be able to build computer-aided tools for any domain!
By the end of this course, you’ll be able to build computer-aided tools for any domain!
logistics

Topics, structure, people
Course overview

program  question

logic

automated reasoning engine
SAT, SMT, model finders & checkers

verifier, synthesizer, fault localizer

logic

program question

Drawing from “Decision Procedures” by Kroening & Strichman
Decision procedures can be rather complex... those that we consider in this book take formulas of different theories as input, possibly mix them (using the Nelson–Oppen procedure – see Chap. 10), decide their satisfiability ("YES" or "NO"), and, if yes, provide a satisfying assignment.

Which Theories? Which Algorithms?

A first-order theory can be considered "interesting", at least from a practical perspective, if it fulfills at least these two conditions:

1. The theory is expressive enough to model a real decision problem. Moreover, it is more expressive or more natural for the purpose of expressing some models in comparison with theories that are easier to decide.

Drawing from “Decision Procedures” by Kroening & Strichman
Course overview

program question

verifier, synthesizer, fault localizer

logic

SAT, SMT, model finders & checkers

study (part I)

build! (part II)

Decision procedures can be rather complex... those that we consider in this book take formulas of different theories as input, possibly mix them (using the Nelson–Oppen procedure – see Chap. 10), decide their satisfiability ("YES" or "NO"), and, if yes, provide a satisfying assignment. Which Theories? Which Algorithms?

A first-order theory can be considered "interesting", at least from a practical perspective, if it fulfills at least these two conditions:

1. The theory is expressive enough to model a real decision problem. Moreover, it is more expressive or more natural for the purpose of expressing some models in comparison with theories that are easier to decide.

Drawing from "Decision Procedures" by Kroening & Strichman
Grading structure

3 individual homework assignments (50%)
- conceptual problems & proofs (Tex)
- implementations in various programming languages
- may discuss problems with others but solutions must be your own

Course project (50%)
- build a computer-aided reasoning tool for a domain of your choice
- teams of 2-3 people strongly encouraged
- see the course web page for timeline, deliverables and other details
Reading and references

Required readings posted on the course web page

• Complete each reading before the lecture for which it is assigned

Recommended text books

• Bradley & Manna, The Calculus of Computation
• Kroening & Strichman, Decision Procedures

Related courses

• Isil Dillig: Automated Logical Reasoning (2013)
• Sanjit Seshia: Computer-Aided Verification (2012)
Advice for doing well in 507

Come to class (prepared)

• Lecture notes are enough to teach from, but not enough to learn from

Participate

• Ask and answer questions

Meet deadlines

• Turn homework in on time
• Start homework and project sooner than you think you need to
• Follow instructions for submitting code (we have to be able to run it)
People

Emina Torlak
PLSE
CSE 596
Wednesdays 1-2

Mert Saglam
Theory
CSE 618
Thursdays 1-2
People

instructor

Emina Torlak
PLSE
CSE 596
Wednesdays 1-2

Mert Saglam
Theory
CSE 618
Thursdays 1-2

TA

students!

Your name
Research area
Survey
Propositional logic: syntax, semantics & proof methods
Syntax of propositional logic

\[(\neg p \land \top) \lor (q \rightarrow \bot)\]
Syntax of propositional logic

Atom

truth symbols: ⊤ ("true"), ⊥ ("false")
propositional variables: p, q, r, …
Syntax of propositional logic

Atom

- Truth symbols: \( \top \) ("true"), \( \bot \) ("false")
- Propositional variables: \( p, q, r, \ldots \)

Literal

- An atom \( \alpha \) or its negation \( \neg \alpha \)

\[(\neg p \land \top) \lor (q \rightarrow \bot)\]
Syntax of propositional logic

**Atom**
- truth symbols: \( \top \) (“true”), \( \bot \) (“false”)
- propositional variables: \( p, q, r, \ldots \)

**Literal**
- an atom \( \alpha \) or its negation \( \neg \alpha \)

**Formula**
- a literal or the application of a logical connective to formulas

\[
(\neg p \land \top) \lor (q \rightarrow \bot)
\]

- \( \neg F \) “not” (negation)
- \( F_1 \land F_2 \) “and” (conjunction)
- \( F_1 \lor F_2 \) “or” (disjunction)
- \( F_1 \rightarrow F_2 \) “implies” (implication)
- \( F_1 \leftrightarrow F_2 \) “if and only if” (iff)
Interpretations of propositional formulas

An interpretation $I$ for a propositional formula $F$ maps every variable in $F$ to a truth value:

$$I : \{ p \mapsto \top, q \mapsto \bot, \ldots \}$$
Interpretations of propositional formulas

An **interpretation** $I$ for a propositional formula $F$ maps every variable in $F$ to a truth value:

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

$I$ is a **satisfying interpretation** of $F$, written as $I \models F$, if $F$ evaluates to $\top$ under $I$.

$I$ is a **falsifying interpretation** of $F$, written as $I \not\models F$, if $F$ evaluates to $\bot$ under $I$. 
Semantics of propositional logic

Base cases:

- $I \models \top$
- $I \not\models \bot$
- $I \models p$ iff $I[p] = \top$
- $I \not\models p$ iff $I[p] = \bot$
Semantics of propositional logic

Base cases:

- $I \models \top$
- $I \not\models \bot$
- $I \models p \iff I[p] = \top$
- $I \not\models p \iff I[p] = \bot$

Inductive cases:
Semantics of propositional logic

Base cases:
• $I \models \top$
• $I \not\models \bot$
• $I \models p \iff I[p] = \top$
• $I \not\models p \iff I[p] = \bot$

Inductive cases:
• $I \models \neg F \iff I \not\models F$
Semantics of propositional logic

**Base cases:**
- \( I \models \top \)
- \( I \not\models \bot \)
- \( I \models p \iff I[p] = \top \)
- \( I \not\models p \iff I[p] = \bot \)

**Inductive cases:**
- \( I \models \neg F \iff I \not\models F \)
- \( I \models F_1 \land F_2 \iff I \models F_1 \) and \( I \models F_2 \)
Semantics of propositional logic

Base cases:
• $I \models \top$
• $I \not\models \bot$
• $I \models p$ iff $I[p] = \top$
• $I \not\models p$ iff $I[p] = \bot$

Inductive cases:
• $I \models \neg F$ iff $I \not\models F$
• $I \models F_1 \land F_2$ iff $I \models F_1$ and $I \models F_2$
• $I \models F_1 \lor F_2$ iff $I \models F_1$ or $I \models F_2$
• $I \models F_1 \rightarrow F_2$ iff $I \not\models F_1$ or $I \models F_2$
• $I \models F_1 \leftrightarrow F_2$ iff $I \models F_1$ and $I \models F_2$, or $I \not\models F_1$ and $I \not\models F_2$
Semantics of propositional logic: example

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

\[ l: \{ p \mapsto \top, q \mapsto \bot \} \]
Semantics of propositional logic: example

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

\[ I: \{ p \mapsto \top, q \mapsto \bot \} \]

\[ I \models F \]
Satisfiability & validity of propositional formulas

\( F \) is **satisfiable** iff \( I \models F \) for some \( I \).

\( F \) is **valid** iff \( I \models F \) for all \( I \).
**Satisfiability & validity of propositional formulas**

- *F* is **satisfiable** iff $I \models F$ for some $I$.

- *F* is **valid** iff $I \models F$ for all $I$.

**Duality** of satisfiability and validity:

- *F* is valid iff $\neg F$ is unsatisfiable.
Satisfiability & validity of propositional formulas

$F$ is **satisfiable** iff $I \models F$ for some $I$.

$F$ is **valid** iff $I \models F$ for all $I$.

**Duality** of satisfiability and validity:

$F$ is valid iff $\neg F$ is unsatisfiable.

If we have a procedure for checking satisfiability, then we can also check validity of propositional formulas, and vice versa.
Techniques for deciding satisfiability & validity

Search

Deduction

SAT solver
Techniques for deciding satisfiability & validity

**Search**

Enumerate all interpretations (i.e., build a truth table), and check that they satisfy the formula.

**Deduction**

**SAT solver**
Enumerate all interpretations (i.e., build a truth table), and check that they satisfy the formula.

Assume the formula is invalid, apply proof rules, and check for contradiction in every branch of the proof tree.
Proof by search (truth tables)

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

$F$: $(p \land q) \rightarrow (p \lor \neg q)$
Proof by search (truth tables)

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

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

Valid.
Proof by deduction (semantic arguments)

Example proof rules:

\[
\begin{align*}
& I \models \neg F & I \models F_1 \land F_2 \\
& I \nvdash F & I \models F_1 \\
& & I \models F_2 \\
& I \nvdash \neg F & I \nvdash F_1 \land F_2 \\
& I \models F & I \nvdash F_1 & I \nvdash F_2
\end{align*}
\]
Proof by deduction (semantic arguments)

Example proof rules:

\[
\begin{align*}
I \not\models \neg F & \quad I \models F_1 \land F_2 \\
I \not\models F & \quad I \models F_1 \\
& \quad I \models F_2 \\
I \not\models \neg F & \quad I \not\models F_1 \land F_2 \\
I \models F & \quad I \not\models F_1 \\
& \quad I \not\models F_2
\end{align*}
\]

\[F: \ p \land \neg q\]
Proof by deduction (semantic arguments)

Example proof rules:

\[
\begin{align*}
I & \not\models \neg F \\
I & \not\models F \\
I & \not\models \neg F \\
I & \not\models F \\
I & \not\models \neg F \\
I & \not\models F \\
I & \not\models \neg F \\
I & \not\models F \\
I & \not\models \neg F \\
I & \models F \\
I & \not\models \neg F \\
I & \not\models F \\
I & \not\models \neg F \\
I & \models F \\
\end{align*}
\]

\[
F: \ p \land \neg q
\]

1. \( I \not\models p \land \neg q \) (assumption)
**Proof by deduction (semantic arguments)**

Example proof rules:

\[ \frac{I ⊭ F}{I ∨ F} \]
\[ \frac{I ⊭ F_1 ∧ F_2}{I ⊭ F_1 \quad I ⊭ F_2} \]

\[ \frac{I ⊭ F}{I ∨ F} \]
\[ \frac{I ⊭ F_1 ∧ F_2}{I ⊭ F_1 \quad I ⊭ F_2} \]

\[ F: \ p \land \neg q \]

1. \( I \not\models p \land \neg q \)  \hspace{1cm} (assumption)
   a. \( I \not\models p \)  \hspace{1cm} (l, \land)
Proof by deduction (semantic arguments)

Example proof rules:

\[
\frac{I \models \neg F}{I \not\models F}
\]

\[
\frac{I \models F_1 \land F_2}{I \models F_1 \quad I \models F_2}
\]

\[
\frac{I \not\models \neg F}{I \models F}
\]

\[
\frac{I \not\models F_1 \quad I \not\models F_2}{I \not\models F_1 \land F_2}
\]

\[
F: \quad p \land \neg q
\]

1. \(I \not\models p \land \neg q\) (assumption)
   
a. \(I \not\models p\) (1, \land)
   
b. \(I \not\models \neg q\) (1, \land)
Proof by deduction (semantic arguments)

Example proof rules:

\[ \frac{I \vdash \neg F}{I \not\models F} \]
\[ \frac{I \vdash F_1 \land F_2}{I \models F_1 \quad I \models F_2} \]
\[ \frac{I \not\models \neg F}{I \models F} \]
\[ \frac{I \not\models F_1 \land F_2}{I \not\models F_1 \quad I \not\models F_2} \]

\( F: \ p \land \neg q \)

1. \( I \not\models p \land \neg q \) (assumption)
   a. \( I \not\models p \) (1, \( \land \))
   b. \( I \not\models \neg q \) (1, \( \land \))
   i. \( I \models q \) (1b, \( \neg \))
Proof by deduction (semantic arguments)

Example proof rules:

\[
\begin{align*}
& I \vdash \lnot F \\
& \quad \vdash F_1 \land F_2 \\
& I \vdash F_1 \\
& I \vdash F_2 \\
& I \not\vdash \lnot F \\
& \quad \vdash F_1 \land F_2 \\
& \quad \vdash F_1 \\& I \not\vdash F_2 \\
& I \vdash F
\end{align*}
\]

\[F: \ p \land \lnot q\]

1. \( I \not\vdash p \land \lnot q \) (assumption)
   a. \( I \not\vdash p \) (1, \land)
   b. \( I \not\vdash \lnot q \) (1, \land)
      i. \( I \vdash q \) (1b, \lnot)

Invalid; \( I \) is a falsifying interpretation.
Semantic judgements

Formulas $F_1$ and $F_2$ are **equivalent**, written $F_1 \iff F_2$, iff $F_1 \leftrightarrow F_2$ is valid.

Formula $F_1$ **implies** $F_2$, written $F_1 \implies F_2$, iff $F_1 \rightarrow F_2$ is valid.
Semantic judgements

Formulas $F_1$ and $F_2$ are **equivalent**, written $F_1 \iff F_2$, iff $F_1 \leftrightarrow F_2$ is valid.

Formula $F_1$ **implies** $F_2$, written $F_1 \implies F_2$, iff $F_1 \rightarrow F_2$ is valid.

$F_1 \iff F_2$ and $F_1 \implies F_2$ are not propositional formulas (not part of syntax). They are properties of formulas, just like validity or satisfiability.
Semantic judgements

Formulas $F_1$ and $F_2$ are **equivalent**, written $F_1 \iff F_2$, iff $F_1 \leftrightarrow F_2$ is valid.

Formula $F_1$ **implies** $F_2$, written $F_1 \implies F_2$, is valid.

If we have a procedure for checking satisfiability, then we can also check for equivalence and implication of propositional formulas.
Summary

Today

• Course overview & logistics
• Review of propositional logic

Next Lecture (by Zach Tatlock)

• Normal forms
• A basic SAT solver

★ Take the course survey
★ Read *Chapter 1* of Bradley & Manna