Lectures 1-2: Overview, Propositional and Predicate Logic

CS536 Science of Programming, Fall 2023
Science of Programming

Specifically, Program Verification
Program Verification

Formally checking that a program is correct

Usually: that it meets a specification

- gives the right answer
- doesn’t take too long
- has the right effects
- has the right security properties

this course (mostly)
Quick Survey

• How many of you have written a program of > 100 lines of code in the last 6 months?
Testing is not enough

... and it matters a lot:

Boeing 737-MAX
2017-2019

Therac-25 radiation therapy machine
1985-1987
Testing is not enough

Even if you cover all code:

• Unexpected inputs
• Unexpected user behavior
• Concurrency errors (e.g., race conditions)
• Changes in code
• Changes in requirements
Application of verification: Be done with your coding homework!
Verification isn’t perfect

• Difficult to get right, even for small programs
• Automated tools can help (but then you have to trust those!)
• Have to get the *specification* right
Easy to get specifications wrong

• What should the spec be?

//Argument: a list of integers
//Returns: ????
function sort (l: int list)
Static types can be seen as a form of verification

- **OCaml**  
  \[
  \text{sort : int list} \rightarrow \text{int list}
  \]
  - Takes an integer list and returns an integer list.
  - Valid: \(\text{sort([8;2;1;6;3])} = [8;2;1;6;3]\)
  - Valid: \(\text{sort([8;2;1;6;3])} = [10;11;12]\)

- **Coq**  
  \[
  \text{sort : forall (l1 : list int), exists (l2: int list),}
  \text{Sorted l2} \lor \text{Permutation l1 l2}
  \]
  - Takes an integer list and returns a sorted permutation of it.
  - Valid: \(\text{sort([8;2;1;6;3])} = [1;2;3;6;8]\)
  - ... and nothing else
Static types can be seen as a form of verification

... but that’s a whole other class
Verification: connecting *logical specs* and *formal semantics*

```c
function f(int x) {
    if (x > 0) {
        y = x * 2;
    } else {
        y = x * -2;
    }
    assert(y >= 0);
}
```

*Formal semantics*: mathematical description of what code does

- If $x > 0$, then $y = + * + = +$
- If $x \leq 0$, then $y = - * - = +$
  or $y = 0 * - = 0$

*How do we know that’s what this code does?*

Well, it’s obvious in this case. But not always (or even defined) in complex languages like C.
Course Outcomes

After taking this class, you should be able to:

• Understand the limits of testing and the importance of verification
• Perform basic verification on programs
• Understand the semantics of programs (including nondeterministic and parallel ones!)
Course Information

• Website: [http://cs.iit.edu/~smuller/cs536-f23/](http://cs.iit.edu/~smuller/cs536-f23/)
  • Schedule, links, notes
  • Check it frequently!

• Blackboard
  • Download and submit assignments
  • Class recordings
Prerequisites

• Officially: CS331 or CS401 with a min. grade of C
• Informally:
  • Familiarity with basic logic
  • Comfort with mathematics, formal notations
  • Some programming experience

• We’ll review some of these concepts quickly today and Wednesday
  • BUT: If you are not at all comfortable with mathematical logic, make sure you learn it this week (not just in class) or consider delaying taking the class
Will there be programming?

• Hard question to answer...
• Will not have to learn a whole new language

• Mostly theory: there will be some proofs
  • Will have to express them formally so they can be checked by computer
  • Proofs about programs: we will be using a small language with assignment, if/then/else, while, etc...
  • May have to write some small programs in this small language
Grading

• 40% Homework assignments (every 1-2 weeks)
  • May not be evenly weighted
• 25% Midterm Exam (Tentatively Oct. 23)
• 35% Final Exam

<table>
<thead>
<tr>
<th>Grade</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>90-100</td>
</tr>
<tr>
<td>B</td>
<td>80-89</td>
</tr>
<tr>
<td>C</td>
<td>70-79</td>
</tr>
<tr>
<td>E</td>
<td>&lt;70</td>
</tr>
</tbody>
</table>

• I may curve exam grades depending on the course averages
Exams

• Midterm: 75 minutes, normal class time
• Final exam during finals period (date set by Registrar)
  • Final exam is cumulative
• Some kind of notes allowed (past semesters: 1-2 sheets of notes)
• Sections 01, 02: In-person
  • If you CANNOT take the exam in person, let me know
• Section 03: Will get an email discussing options
  • Preferred: take with the in-person students.
  • Also possible: take somewhere else with a proctor
Late Days

• 8 late days per student
• Each late day extends the deadline 24 hours
• Can use $\leq 2$ per assignment
  • Can’t use on exams
• After late days used up: 10% penalty per day late
• No work accepted $>2$ days late without instructor approval
Academic Honesty

• All submitted work (homework and exams) is to be your own individual work unless specified otherwise.

• Specifically prohibited (but this list isn’t exhaustive):
  • Sharing answers with other students
  • Looking online for answers
  • Generative AI (e.g., ChatGPT)

• Specifically permitted:
  • Getting help from TAs or instructor
  • Getting help from the ARC or other official university tutoring resources
    • If you want to use an outside tutor, let me know first
Academic Honesty

• Penalties (for every violation):
  • Zero on the homework or exam
  • Report to academic honesty
    • May result in university-level sanctions after first report
Course Staff

- **Instructor:** Stefan Muller
- **TAs:** Chaoqi Ma, Gagan Beerappa

- Office hours (info including links will be posted):
  - Monday 2pm-3pm SB 218E (Stefan)
  - Tuesday 2pm-3pm SB 004 (Chaoqi)
  - Wednesday 2pm-3pm Google Meet (Gagan)
  - Thursday 10am-11am Zoom (Stefan) AND 2pm-3pm Zoom (Chaoqi)
  - Friday 2pm-3pm Google Meet (Gagan)

- We are here to answer your questions! Really! Yes, all of us!
Other ways to get help

- Discord: IIT CS server, cs536 channel
  - If you’re not on it, we’ll send an invitation

- Academic Resource Center (ARC): www.iit.edu/arc
  - FREE subject matter tutoring and academic coaching

<table>
<thead>
<tr>
<th>Question</th>
<th>Discord</th>
<th>Office Hours</th>
<th>Email</th>
<th>ARC</th>
</tr>
</thead>
<tbody>
<tr>
<td>General questions about lectures, logistics, etc.</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>General discussion, clarifications, about HW questions</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specific questions about your HW answers</td>
<td></td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>More in-depth personal tutoring</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Personal matters (accommodations, other requests, etc.)</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Attendance/Sections

• Section 01: In-person
• Section 02: In-person, PhD section
• Section 03: Online (lectures recorded)

• Also:
  • Sections 04, 06: In-person with other instructors
  • Section 05: Online with other instructor
PhD Qualifier Section

• A sufficiently high grade in CS536 meets the requirements for the written qualifier for CS PhDs.
  • **Only if you are in section 02.**
    • If you are taking this class to meet this requirement and are not in section 02, talk to me or switch this ASAP.

• Section 02 is an in-person section. If you’re a PhD student taking 536 for the qualifier but can’t attend in person, let me know.
Announcements

• Blackboard fixed
• Office Hours times/links are up on Blackboard
  • (times on course website)
• HW1 will be posted today
  • On Blackboard
  • Due 9/7, 11:59 PM (remember: can use up to 2 late days)
  • Submit on Blackboard
  • Start early, make sure you can do it
Syntax and Semantics and Equality

• Syntax: How to write down a “program”
  • Syntactic Equality (≡): written the same (up to, e.g., parentheses)
  • $2 + 2 - 3 \equiv 2 + 2 - 3 \equiv (2 + 2) - 3$
  • $2 + 2 - 3 \not\equiv 1$
  • $1 + 2 \not\equiv 2 + 1$

• Semantics: What a “program” “means”
  • Semantic Equality (≡): has the same meaning
  • $2 + 2 - 3 = (2 + 2) - 3 = 4 - 3 = 1$
  • $1 + 2 = 2 + 1$
Propositional Logic

• “Atomic” propositions: variables that can be true (T) or false (F)
  • P, Q

• Connectives: make larger propositions p, q, φ, ψ
  • Negation: \( \neg p \) (not \( p \))
  • Conjunction: \( p \land q \) (p and q)
  • Disjunction: \( p \lor q \) (p or q)
  • Conditional: \( p \rightarrow q \) (p implies q, if p then q)
  • Biconditional: \( p \leftrightarrow q \) (p iff q, p if and only if q)

• Precedence (order of operations): in the above order
  \[
p \land q \rightarrow r \lor \neg q \leftrightarrow s \equiv [(p \land q) \rightarrow (r \lor (\neg q))] \leftrightarrow s
  \]
The semantics of a proposition are their truth values in different states

State $\sigma$: Assignment of truth values (T, F) to proposition variables
Written, e.g., $\{P = T, Q = F\}$
Only one assignment per variable: $\{P = T, P = F\}$
Only assigns to variables: $\{P \lor Q = T\}$
A state fitting these requirements is well-formed (opp. ill-formed)

$\sigma \models p$: $p$ “satisfied” (true) in state $\sigma$
Truth value of propositions determined by truth tables

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P → Q</th>
<th>P ↔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

• ∧, ∨ are commutative and associative: $P \land Q = Q \land P$  
  $P \land (Q \land R) = (P \land Q) \land R$

• $→$ is not commutative or associative:  
  $F \to T \neq T \to F$  
  $(F \to T) \to F \neq F \to (T \to F)$

• $↔$ is commutative and associative:  
  $P \leftrightarrow Q = Q \leftrightarrow P$  
  $(P \leftrightarrow Q) \leftrightarrow R = P \leftrightarrow (Q \leftrightarrow R)$
Some more facts about conditionals

For a conditional $P \rightarrow Q$:

• The inverse $\neg P \rightarrow \neg Q$ does not have the same truth value

• The converse $Q \rightarrow P$ does not have the same truth value
  • (But is the same as the inverse)

• The contrapositive $\neg Q \rightarrow \neg P$ has the same truth value
To determine truth value, a state needs to be *proper* for the proposition

- Proper: defines truth values for all variables in the proposition

Proposition: \( P \land Q \rightarrow R \lor \neg Q \leftrightarrow S \)

**Proper:**
- \( \{P = T, Q = F, R = F, S = T\} \)
- \( \{Q = T, P = F, S = F, R = T\} \)
- \( \{Q = T, P = F, S = F, R = T, T = F\} \)

**Improper:**
- \( \{P = T, Q = F, R = F\} \)
- \( \{P = F, S = F\} \)
For a well-formed and proper state, a proposition is satisfied or unsatisfied

- \( \{P = T; Q = F; R = F\} \models (P \land Q) \rightarrow R? \)
- \( \{P = T; Q = F; R = T\} \models (P \land Q) \rightarrow R? \)
- \( \{P = T; Q = F; R = F\} \models (P \lor Q) \rightarrow R? \)
A proposition can be a tautology, contradiction or contingency

Assume $\sigma$ is well-formed and proper

• Tautology: $\sigma \models p$ for all $\sigma$ (Also write just $\models p$)
• Contradiction: $\sigma \not\models p$ for all $\sigma$ (Equivalent: $\models \neg p$)
  • Note that this is not the same as $\not\models p$ (that just says $p$ is not a tautology)
• Contingency: There exist $\sigma_1$ and $\sigma_2$ such that $\sigma_1 \models p$ and $\sigma_2 \not\models p$
  (Equivalent: $\not\models \neg p$ and $\not\models p$)
Logical implication $\Rightarrow$

$P \Rightarrow Q$ if whenever $P$ is true, so is $Q$ ("$P$ implies $Q$")

Note: this is not the same as $P \rightarrow Q$:

• They’re related: $P \Rightarrow Q$ means $\models P \rightarrow Q$

• We write $\rightarrow$ in propositions, we use $\Rightarrow$ to talk about propositions
  • (like how we put + in mathematical expressions, we use = to talk about them)
Logical equivalence ⇔

P ⇔ Q means P ⇒ Q and Q ⇒ P

Semantic equality (=) on logical propositions
Some useful facts

Commutativity: $p \lor q \iff q \lor p$

$\quad p \land q \iff q \land p \quad (p \iff q) \iff (q \iff p)$

Associativity: $(p \lor q) \lor r \iff p \lor (q \lor r)$

$(p \land q) \land r \iff p \land (q \land r)$

Distributivity/Factoring

$(p \lor q) \land r \iff (p \land r) \lor (q \land r)$

$(p \land q) \lor r \iff (p \lor r) \land (q \lor r)$

Transitivity [Note: $\Rightarrow$, not $\iff$ here]

$(p \rightarrow q) \land (q \rightarrow r) \Rightarrow (p \rightarrow r)$

$(p \iff q) \land (q \iff r) \Rightarrow (p \iff r)$

Identity: $p \land T \iff p \land p \iff p$

Idempotency: $p \lor p \iff p$ and $p \land p \iff p$

Domination: $p \lor T \iff T$ and $p \land F \iff F$

Absurdity: $(F \rightarrow p) \iff T$

Contradiction: $p \land \neg p \iff F$

Excluded middle: $p \lor \neg p \iff T$

Double negation: $\neg \neg p \iff p$

DeMorgan’s Laws

$\neg (p \land q) \iff (\neg p \lor \neg q)$

$\neg (p \lor q) \iff (\neg p \land \neg q)$

Defn. of $\rightarrow$ and $\iff$ $(p \rightarrow q) \iff (\neg p \lor q)$

$(p \iff q) \iff (p \rightarrow q) \land (q \rightarrow p)$
-(p \rightarrow q) \Rightarrow (p \land \neg q)

Commutativity \: p \lor q \Leftrightarrow q \lor p
\: p \land q \Leftrightarrow q \land p \: (p \leftrightarrow q) \Leftrightarrow (q \leftrightarrow p)

Associativity \: (p \lor q) \lor r \Leftrightarrow p \lor (q \lor r)
\: (p \land q) \land r \Leftrightarrow p \land (q \land r)

Distributivity/Factoring
\: (p \lor q) \land r \Leftrightarrow (p \land r) \lor (q \land r)
\: (p \land q) \lor r \Leftrightarrow (p \lor r) \land (q \lor r)

Transitivity [Note: \Rightarrow, not \Leftrightarrow here]
\: (p \rightarrow q) \land (q \rightarrow r) \Rightarrow (p \rightarrow r)
\: (p \leftrightarrow q) \land (q \leftrightarrow r) \Rightarrow (p \leftrightarrow r)

Identity: \: p \land T \Leftrightarrow p \: and \: p \lor F \Leftrightarrow p

Idempotentcy: \: p \lor p \Leftrightarrow p \: and \: p \land p \Leftrightarrow p

Domination: \: p \lor T \Leftrightarrow T \: and \: p \land F \Leftrightarrow F

Absurdity: \: (F \rightarrow p) \Leftrightarrow T

Contradiction: \: p \land \neg p \Leftrightarrow F

Excluded middle: \: p \lor \neg p \Leftrightarrow T

Double negation: \: \neg \neg p \Leftrightarrow p

DeMorgan’s Laws \: \neg(p \land q) \Leftrightarrow (\neg p \lor \neg q)
\: \neg(p \lor q) \Leftrightarrow (\neg p \land \neg q)

Defn. of \rightarrow and \Leftrightarrow \: (p \rightarrow q) \Leftrightarrow (\neg p \lor q)
\: (p \leftrightarrow q) \Leftrightarrow (p \rightarrow q) \land (q \rightarrow p)
Commutativity \( p \lor q \iff q \lor p \)
\( p \land q \iff q \land p \) (\( p \leftrightarrow q \iff q \leftrightarrow p \))

Associativity
\( (p \lor q) \lor r \iff p \lor (q \lor r) \)
\( (p \land q) \land r \iff p \land (q \land r) \)

Distributivity/Factoring
\( (p \lor q) \land r \iff (p \land r) \lor (q \land r) \)
\( (p \land q) \lor r \iff (p \lor r) \land (q \lor r) \)

Transitivity [Note: \( \Rightarrow \), not \( \iff \) here]
\( (p \rightarrow q) \land (q \rightarrow r) \Rightarrow (p \rightarrow r) \)
\( (p \leftrightarrow q) \land (q \leftrightarrow r) \Rightarrow (p \leftrightarrow r) \)

Identity: \( p \land T \iff p \) and \( p \lor F \iff p \)

Idempotency: \( p \lor p \iff p \) and \( p \land p \iff p \)

Domination: \( p \lor T \iff T \) and \( p \land F \iff F \)

Absurdity: \( (F \rightarrow p) \iff T \)

Contradiction: \( p \land \neg p \iff F \)

Excluded middle: \( p \lor \neg p \iff T \)

Double negation: \( \neg \neg p \iff p \)

DeMorgan’s Laws
\( \neg(p \land q) \iff (\neg p \lor \neg q) \)
\( \neg(p \lor q) \iff (\neg p \land \neg q) \)

Defn. of \( \rightarrow \) and \( \iff \)
\( (p \rightarrow q) \iff (p \lor q) \land (q \rightarrow p) \)

\((r \rightarrow s) \land r \Rightarrow s \) (“Modus ponens”)
$T \Rightarrow P \land \neg (Q \land R) \Rightarrow ((Q \land R) \Rightarrow \neg P)$
Announcements

• New TA: Param Modi
  • Office Hours: Tuesday/Thursday 11:30am-12:30pm (Online)

• HW1 is up on Blackboard.
  • Covers material through part of today’s lecture
More facts

- If $\models p$ and $\models q$ then $\models p \land q$
- If $\models p$ then $\models p \lor q$ and $\models q \lor p$ (for any $q$)
- If $\models p \land q$ then $\models p$ and $\models q$
- If $\models p \rightarrow r$ and $\models q \rightarrow r$ and $\models p \lor q$ then $\models r$
Predicate (First-Order) Logic extends Prop. Logic with values in a domain

• (e.g. the integers)

• We’ll also use variables like \( x \) to hold integers (or values of whatever domain we’re using)

• Predicate: a function from values or variables in the domain to \( T \) or \( F \)
  • e.g. isEven(\( x \)), Greater(\( x, 0 \)), Greater(\( x, y \))
    “Syntactic sugar”: \( x > 0 \), \( x > y \)
We can use predicates with all the existing connectives

• $\text{Greater}(x, y) \lor \text{Greater}(y, x) \lor \text{Equal}(x, y)$
• $\text{Greater}(x, y) \lor \text{Equal}(x, y)$
• $\text{Greater}(x, y) \rightarrow \text{Greater}(x, y) \lor \text{Equal}(x, y)$
• $\text{Greater}(x, y) \lor \text{Equal}(x, y) \leftrightarrow \neg \text{Greater}(y, x)$
States can now have integer vars too

- \{x = 5, y = 5\} \models \text{Greater}(x, y) \lor \text{Equal}(x, y)
- \{x = 4, y = 5\} \not\models \text{Greater}(x, y) \lor \text{Equal}(x, y)
- \{x = 5, y = 5, P = T\} \models (\text{Greater}(x, y) \lor \text{Equal}(x, y)) \land P
Quantifiers introduce variables

- \( \forall x \in \mathbb{Z}. p \) *(for all x, p)*
- \( \exists x \in \mathbb{Z}. p \) *(there exists x such that p)*
- (may omit the domain if clear)

- \( \models \forall x. \forall y. \text{Greater}(y, x) \lor \text{Greater}(x, y) \lor \text{Equal}(x, y) \)
- \( \models \forall x. \forall y. \text{Greater}(x, y) \lor \text{Equal}(x, y) \leftrightarrow \neg \text{Greater}(y, x) \)
- \( \models \forall x. \exists y. \text{Greater}(y, x) \)
- \( \models \neg \exists x. \text{Greater}(x, 2) \land \text{isPrime}(x) \land \text{isEven}(x) \)
Equivalence with quantifiers gets a little tricky

• $\forall x. P(x) = \forall y. P(y)$ because $\forall x. P(x) \iff \forall y. P(y)$

• Is $\forall x. P(x) \equiv \forall y. P(y)$?
  • For now, let’s say no.
  • But there are good reasons to consider them equivalent in more-than-just-semantic ways. We may discuss this later.
DeMorgan’s Laws for quantifiers

- $\neg \exists x. P \iff \forall x. \neg P$
- $\neg \forall x. P \iff \exists x. \neg P$

$\neg \exists x. \text{Greater}(x, 2) \land \text{isPrime}(x) \land \text{isEven}(x)$
$\iff \forall x. \neg \text{Greater}(x, 2) \lor \neg \text{isPrime}(x) \lor \neg \text{isEven}(x)$
∀x. ∃y. Greater(y, x) ⇔ ¬∃x. ∀y. ¬Greater(y, x)
How would we actually go about proving $\models \forall x. \exists y. \text{Greater}(y, x)$?

Formal systems for this kind of proof are complicated (we’d need to know the semantics of Greater), but here’s an idea:

- To prove $\models \forall x. p(x)$, $p(x)$ must hold *regardless* of the choice of $x$.
- To prove $\models \exists x. p(x)$, come up with a *witness*: a value of $x$ such that $p(x)$ holds.
Examples

\( \forall x \in \mathbb{Z} . x \neq 0 \rightarrow x \leq x^2 \)  
\( \forall x \in \mathbb{Z} . x \neq 0 \land x \geq x^2 \)

True

True (use 1 as a witness)

\( x > 0 \rightarrow \exists y . y^2 < x \)  
\( x > 0 \rightarrow y^2 < x \)

Tautology (0 works as a witness regardless of choice of x)

Contingency

\( \exists y . (y < 0 \land y > x^2) \)

Contradiction (false for every choice of x)
We can define our own predicates

• e.g. Positive(x) = Greater(x, 0) \land \neg Equal(x, 0)

• The body should be a proposition over the parameters to the predicate function.
  • e.g. **not** square(x) = x * x
  • but: square(x, y) = (y = x * x)
Predicates should be simple

- For an array a, \( \text{AllPositive}(a, m, n) \), should mean that \( a[m], ..., a[n] \) are all positive.

- First try: \( \text{AllPositive}(a, m, n) = \text{Positive}(a[m]) \land ... \land \text{Positive}(a[n]) \)

- Second try: maybe a loop?

- Fine in a regular programming language, but the purpose of our predicates is debugging programs
  - No point if our predicates are as hard to debug as the programs!

- \( \text{AllPositive}(a, m, n) = \forall i, (m \leq i \land i \leq n) \rightarrow \text{Positive}(a[i]) \)
Sorted as a predicate

- Sorted(a, m, n): a[m], ..., a[n] are in sorted order
  - (i.e. a[m] ≤ a[m+1] ≤ ... ≤ a[n])
  - (i.e., a[m] ≤ a[m+1] and a[m+1] ≤ a[m+2] and...)
  - (i.e., for all i, a[i] ≤ a[i+1])

- Sorted(a, m, n) = ∀i, (m ≤ i ∧ i < n) → a[i] ≤ a[i + 1]