Logical Evidence

\[ x \notin \Gamma \quad \frac{[\Gamma, x] \ p}{[\Gamma] \ \forall x, \ p} \quad \frac{[\Gamma] \ e}{[\Gamma] \ \exists x, \ p} \]

What about evidence for other connectives?

**Statement** \( p \)  
Convert \( p \) into **prefix form**; \( p' \) has no quantifiers

**Statement** \( Qx_1, \ldots, Qx_n, p' \)  
Convert \( p' \) into **conjunctive form**

**Statement** \( Qx_1, \ldots, Qx_n, (a_1 \lor a_2 \lor \ldots) \land (b_1 \lor \ldots) \land \ldots \)
Axioms

Set of basic facts you use **to prove other facts**

\[[\Gamma] a_1, a_2, \ldots \vdash p\]

**Axioms**

Axioms can be **quantified**, with rules:

\[\begin{align*}
[\Gamma] e & \quad \frac{[\Gamma] a[x := e] \vdash p}{[\Gamma] \forall x, a \vdash p} \\
[\Gamma] & \quad \frac{x \notin \Gamma}{[\Gamma] \exists x, a \vdash p}
\end{align*}\]
Domain Facts

Basic facts **imply** other facts!  

Abstract block to boolean

\[
x \times 0 = 0 \\
x = x \\
x < 0 \lor x = 0 \lor 0 < x \\
x < 0 \land x < 0 \rightarrow x \times 0 < x \times x \\
0 < x \land 0 < x \rightarrow x \times 0 < x \times x \\
x = 0 \land x = x \rightarrow x \times x = 0 \times 0 \\
x \times x = 0 \lor 0 < x \times x
\]
Whose Axioms?

Axioms come with the **theory**, not the problem
Responsibility of **logic designer**, not the prover

Prover questions

What’re the axioms?
Which to use?

Logician questions

Are they correct?
Are they useful?

No easy answers!
Class Progress

Logical reasoning

Program logics

Static analysis

First-order Logic

Decision Procedures

Boolean logic

Syntax

Proof

Theory
First-order Theories

How do we reason about equality?

Equality as substitution

How do we reason about integers?

Many choices of axioms

Limited complexity in software verification problems

How do we reason about arrays?

Defining new operations, gluing together theories
Equality
The true meaning of identity
Equality Axioms

How do you **prove** two things are equal?

**Reflection**
\[ \forall x, x = x \]

**Symmetry**
\[ \forall x, \forall y, x = y \rightarrow y = x \]

**Transitivity**
\[ \forall x, \forall y, \forall z, (x = y \land y = z) \rightarrow x = z \]

Plus, **domain-specific** axioms

**Commutativity**
\[ \forall x, \forall y, x + y = y + x \]

**Read-over-write**
\[ \forall a, \forall i, \forall x, a[i := x][i] = x \]
Using Equality

What does equality tell us about $x$ and $y$?

$$x = y$$

It tells us that $x$ and $y$ have the **same value**

Which means we can replace $x$ by $y$ in any expression

$$f(x) = f(y)$$
Functional Equality

Fundamental axiom of equality as substitution

Axiom schema, a “menu” for axioms

\[ \forall x, \forall y, x = y \rightarrow f(x) = f(y) \]
\[ \forall x, \forall y, x = y \rightarrow P(x) \leftrightarrow P(y) \]

Extends to multi-argument functions and relations

\[ \forall x_1, \forall y_1, \ldots, x_1 = y_1 \land x_2 = y_2 \land \ldots \rightarrow \]
\[ f(x_1, x_2, \ldots) = f(y_1, y_2, \ldots) \]
\[ R(x_1, x_2, \ldots) \leftrightarrow R(y_1, y_2, \ldots) \]
Two Types of Axioms

**Construction**

<table>
<thead>
<tr>
<th>Reflection</th>
<th>Symmetry</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x, x = x$</td>
<td>$\forall x, \forall y, x = y \rightarrow y = x$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x, \forall y, \forall z, (x = y \land y = z) \rightarrow x = z$</td>
</tr>
</tbody>
</table>

**Use**

<table>
<thead>
<tr>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x_1, \forall y_1, \ldots, x_1 = y_1 \land x_2 = y_2 \land \ldots \rightarrow$</td>
</tr>
</tbody>
</table>

| f(x_1, x_2, \ldots) = f(y_1, y_2, \ldots) | R(x_1, x_2, \ldots) \leftrightarrow R(y_1, y_2, \ldots) |
Example Proof

Symmetry **can be proven** from substitution

\[ \forall x, \forall y, x = y \rightarrow P(x) \leftrightarrow P(y) \text{ with } P(z) := z = x \]

\[ \forall x, x = x, \forall x, \forall y, x = y \rightarrow (x = x \leftrightarrow y = x) \vdash \forall x, \forall y, x = y \rightarrow y = x \]
Example Proof

Symmetry can be proven from substitution

<table>
<thead>
<tr>
<th>A</th>
<th>∧</th>
<th>B → (A ↔ C)</th>
<th>→</th>
<th>B → C</th>
</tr>
</thead>
<tbody>
<tr>
<td>[x, y]</td>
<td></td>
<td>x = x , x = y → (x = x ↔ y = x)</td>
<td>⊢</td>
<td>x = y → y = x</td>
</tr>
<tr>
<td>[x, y]</td>
<td></td>
<td>∀x, x = x , ∀x, ∀y, x = y → (x = x ↔ y = x)</td>
<td>⊢</td>
<td>x = y → y = x</td>
</tr>
<tr>
<td>[ ]</td>
<td></td>
<td>∀x, x = x , ∀x, ∀y, x = y → (x = x ↔ y = x)</td>
<td>⊢</td>
<td>∀x, ∀y, x = y → y = x</td>
</tr>
</tbody>
</table>

Proof requires instantiating axioms, quantifiers

Major challenge for automatic proof search
Is it Enough?

Reflexivity + substitution axioms prove any equality fact

Proof uses model theory, mathematical study of proof systems

However, challenging for computers

Need to invent functions and relations for axioms

Non-quantified statements easy to prove (next Tue!)

<table>
<thead>
<tr>
<th>Quantified statements</th>
<th>Non-quantified statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impossible</td>
<td>O(n log n)</td>
</tr>
</tbody>
</table>
Integers

Three ways to reason about numbers
History

Induction can prove many arithmetic facts

We could have some axioms for arithmetic

I have a full second-order axiom system
History

Induction can prove many arithmetic facts. We could have some axioms for arithmetic. I have a full second-order axiom system. Guys check out my book: *Arithmetices principia, nova metodo exposita*. It has first-order axioms. Also I invented some symbols: $\land, \lor, \subset, \in$. 
The Axioms

Axioms for equality (reflexivity, substitution)

Injectivity
\[ \forall x, \forall y, x + 1 = y + 1 \rightarrow x = y \]

Discrimination
\[ \forall x, x + 1 \neq 0 \]

Definition of the \( x + 1 \) operation:

Definition of addition and multiplication:

Addition
\[ \forall x, x + 0 = x \]
\[ \forall x, \forall y, x + (y + 1) = (x + y) + 1 \]

Multiplication
\[ \forall x, x \times 0 = 0 \]
\[ \forall x, \forall y, x \times (y + 1) = (x \times y) + x \]
The Axioms

Axioms for equality (reflexivity, substitution)

Definition of the $x + 1$ operation

Definition of addition and multiplication

The axiom of **induction**:

$$P(0) \land \left( \forall x, P(x) \implies P(x + 1) \right) \implies \left( \forall x, P(x) \right)$$

First:

assume $0 \leq n$
Induction

Let's prove \( \forall n, \exists k, n = 2 \times k \lor n = 2 \times k + 1 \)

\[
P(n) := \exists k, n = 2 \times k \lor n = 2 \times k + 1
\]

\[
P(0) := \exists k, 0 = 2 \times k \lor 0 = 2 \times k + 1
\]

\[
k := 0
\]

\[
0 = 2 \times 0 \lor 0 = 2 \times 0 + 1
\]
Induction

Let’s prove $\forall n, \exists k, n = 2 \times k \lor n = 2 \times k + 1$

$P(n) := \exists k, n = 2 \times k \lor n = 2 \times k + 1$

$\exists k, n = 2 \times k \lor n = 2 \times k + 1$

$\exists k, n + 1 = 2 \times k \lor n + 1 = 2 \times k + 1$

$k := k$

$\forall$

$\exists k', n + 1 = 2 \times k' \lor n + 1 = 2 \times k' + 1$

$k' := k + 1$
Let's prove \( \forall n, \exists k, n = 2 \times k \lor n = 2 \times k + 1 \)

\[
P(n) := \exists k, n = 2 \times k \lor n = 2 \times k + 1
\]
Naturals to Integers

Can reason about integers in terms of natural numbers
Each integer: difference of a pair of natural numbers

\( \forall x, \exists y, \exists z, x < y - z \rightarrow x \times y - z < z \)

Move negative terms across inequalities
Result: natural number equation with twice the variables
Using the Axioms

The Peano axioms are **powerful** but **complex**
Difficult to come up with $P$ values for induction

<table>
<thead>
<tr>
<th>Quantified statements</th>
<th>Non-quantified statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impossible</td>
<td>Impossible</td>
</tr>
</tbody>
</table>

**Full power not needed** for verification
Number theory, provability, complex identities don’t come up
Find **fragments** of the axioms that computers can handle
Variant Axioms

**Full power not needed** for verification

Find **fragments** of the axioms that computers can handle

<table>
<thead>
<tr>
<th>No induction</th>
<th>Robinson Arithmetic</th>
<th>Impossible</th>
</tr>
</thead>
<tbody>
<tr>
<td>No multiplication*</td>
<td>Presberger Arithmetic</td>
<td>O(exp(exp(exp(n)))))</td>
</tr>
<tr>
<td>No multiplication, quantifiers*</td>
<td>Linear Integer Arithmetic</td>
<td>O(exp(exp(n)))</td>
</tr>
</tbody>
</table>

* Of variables; multiplication by constants is repeated addition

Weak fragment **sufficient for verification** of most code

Later in the class: what about code where it's not sufficient?
Course Updates
Assignment 1 Grading
Assignment 1 grading done

Common issues: late submissions, input / output format
Please put your name in file name; it helps grading

Please review your grade and comments
Let us know if we missed something

10 points given for bonus problem

Assignment 2 posted, due next Thursday
Solve KenKen problems with Z3; compare to MiniSat
Arrays
Combining multiple theories
Theory of Arrays

Arrays have a value for every index

**Sorts**
- Int
- Array

**Constants**
- \( n : \text{Int} \)

**Functions**
- \( -\text{Int} : \text{Int} \)
- \( \text{Int} + \text{Int} : \text{Int} \)
- \( \text{Int} \times \text{Int} : \text{Int} \)
- \( \text{Array}[\text{Int}] : \text{Int} \)
- \( \text{len} (\text{Array}) : \text{Int} \)
- \( \text{Array}[\text{Int} := \text{Int}] : \text{Array} \)

**Relations**
- \( \text{Int} = \text{Int} \)
- \( \text{Int} < \text{Int} \)
- \( \text{Int} \in \text{Array} \)

Array reasoning requires value and index reasoning

Depend on axioms for values and axioms for indices
Axioms of Arrays

Define one theory in terms of another

Axioms for equality (reflexivity, substitution)

Axioms for integers (pick your favorite)

Definition of “array set”

\[ \forall a, \forall x, \forall i, \forall j, i = j \land j < \text{len}(a) \rightarrow a[i := x][j] = x \]

\[ \forall a, \forall x, \forall i, \forall j, i \neq j \land j < \text{len}(a) \rightarrow a[i := x][j] = a[j] \]

\[ \forall a, \forall i, \forall x, \text{len}(a[i := x]) = \text{len}(a) \]

Definition of “array contains”

\[ \forall a, \forall x, x \in a \leftrightarrow \exists i, i < \text{len}(a) \land a[i] = x \]
Example

Know

∀ i, ∀ j, i < j → left2[i] < left2[j] *

∀ i, ∀ j, i < j → right2[i] < right2[j] *

∀ i, ∀ j, left2[i] < right2[j]

Want

∀ i, ∀ j, i < j → (left2 + right2)[i]

< (left2 + right2)[j] *
Defining Append

Add axioms to define array append

\[ \forall a, \forall b, \forall i, i < \text{len}(a) \rightarrow (a + b)[i] = a[i] \]

\[ \forall a, \forall b, \forall i, \neg(i < \text{len}(a)) \rightarrow (a + b)[i] = b[i - \text{len}(a)] \]

\[ \forall a, \forall b, \text{len}(a + b) = \text{len}(a) + \text{len}(b) \]

Later, we’ll prove these from an implementation
Mixed Theories

First-order logic is **common framework** for logics

Can **freely add** sorts, functions, relations, axioms

Convenient to **mix + combine** theories

Solvers handle **first-order core**, plug in domain models

Next time: **separating** a first-order problem into domains
Types of Axioms

**Construction + use** for relations

Also called “introduction” and “elimination” rules

**Definitions** for functions and relations

In terms of other, more basic, functions / relations

**Induction rules** for data structures

Involves injectivity, discrimination, and induction axioms
Logic: A Summary

How do we state and prove specifications?
First-order Logic

First-order logic is **common framework** for logics

\[ p, q := \neg p \mid p \lor q \mid p \land q \mid \forall v, p \mid \exists v, p \]

Can add **domain-specific** constructs

- **Sorts, constants, functions, and relations** define the syntax
- **Interpretations** of each define the meaning
- **Axioms** allow proving first-order statements

Domains can be **mixed** to solve complex problems
Proofs

Proofs provide **compact evidence** for a first-order fact

\[ \vdash A_1, A_2, \ldots, A_n \quad \text{Fact} \]

Quantifier rules for manipulating facts & axioms

\[
\begin{align*}
[\Gamma] A & \vdash \forall x, p & [\Gamma] A & \vdash \exists x, p & [\Gamma] \exists x, A & \vdash p & [\Gamma] \forall x, A & \vdash p
\end{align*}
\]

Replace identical expressions with boolean variables
Next Steps

Know how to construct proofs **manually**
Choose axioms, remove quantifiers, find resolution proof

**Automated proofs** necessary for verification

Universal solver for first-order logic **impossible**
Need **domain-specific** reasoning
**Separate problem** into individual domains
**Combine DPLL** with domain-specific solvers
Class Progress

Logical reasoning
Program logics
Static analysis
First-order Logic
Decision Procedures
Boolean logic
Syntax
Proof
Theory
Class Progress

Logical reasoning

Program logics

Static analysis

First-order Logic

Decision Procedures

Mixing Theories

Equality

Integers

Arrays
Next class:
Nelson-Oppen

To do:
- Course feedback
- Read Chapter 3
- Assignment 2
First-order Theories

How do we reason about equality?
   Equality as substitution

How do we reason about integers?
   Many choices of axioms
   Limited complexity in software verification problems

How do we reason about arrays?
   Defining new operations, gluing together theories
SEPARATION

TWO DOMAINS

TWO SOLVERS
ARRANGEMENTS

INTERFACE

BETWEEN

DOMAINS
BACKTRACKING

DPLL(T)
Next class:
Nelson-Oppen

To do:
- Course feedback
- Read Chapter 3
- Assignment 2
;;; Axioms and definitions
(declare-fun len ((Array Int Int)) Int)

(define-fun index ((a (Array Int Int)) (i Int)) Bool
(and (<= 0 i) (< i (len a))))

;; Declare the existence of the "append" function
(declare-fun append ((Array Int Int) (Array Int Int)) (Array Int Int))

;; Axiom 1: select from the left half of an appended array
(assert (forall ((a (Array Int Int)) (b (Array Int Int)) (i Int))
(=> (index (append a b) i)
(< i (len a))
(= (select (append a b) i) (select a i))))))

;; Axiom 2: select from the right half of an appended array
(assert (forall ((a (Array Int Int)) (b (Array Int Int)) (i Int))
(=> (index (append a b) i)
(>= i (len a))
(= (select (append a b) i) (select b (- i (len a)))))))

;; Axioms 3: length of an appended array
(assert (forall ((a (Array Int Int)) (b (Array Int Int)))
(= (len (append a b)) (+ (len a) (len b)))))

;; Definition of an array being sorted
(define-fun sorted ((a (Array Int Int))) Bool
(forall ((i Int) (j Int))
(=> (index a i) (index a j)
(< i j)
(< (select a i) (select a j))))))

;; Definition of two arrays being partitioned
(define-fun partitioned ((a (Array Int Int)) (b (Array Int Int))) Bool
(forall ((i Int) (j Int))
(=> (index a i) (index b j) (< (select a i) (select b j))))))

;;;; Actual problem

;;; "left2" and "right2" are sorted arrays
(declare-const left2 (Array Int Int))
(declare-const right2 (Array Int Int))
(assert (sorted left2))
(assert (sorted right2))

;;; "left2" is less than "right2"
(assert (partitioned left2 right2))
(assert (not (sorted (append left2 right2))))
(check-sat)

Run with: z3 -smt2 file