First-order Theories

Specifications section, **Logic** topic, **Lecture 5**

Pavel Panchekha

CS 6110, U of Utah

21 January 2020

Logical Evidence

$$x \notin \Gamma \; \frac{[\Gamma, x]\; p}{[\Gamma]\; \forall x, p} \qquad\qquad [\Gamma]\; e \; \frac{[\Gamma]\; p[x := e]}{[\Gamma]\; \exists x, p}$$

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 $Qx_1, \ldots, Qx_n, (a_1 \vee a_2 \vee \ldots) \wedge (b_1 \vee \ldots) \wedge \ldots$

Axioms

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

$$[\Gamma]\ \underbrace{a_1, a_2, \ldots}_{\textbf{Axioms}} \vdash p$$

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

$$[\Gamma]\ e\ \frac{[\Gamma]\ a[x := e] \vdash p}{[\Gamma]\ \forall x, a \vdash p} \qquad x \notin \Gamma\ \frac{[\Gamma, x]\ a \vdash p}{[\Gamma]\ \exists x, a \vdash p}$$

Domain Facts

Basic facts **imply** other facts!

Abstract block to boolean

$$x \times 0 = 0 \qquad \boxed{x = x}$$

$$\boxed{x < 0} \vee \boxed{x = 0} \vee \boxed{0 < x}$$

$$\boxed{x < 0} \wedge \boxed{x < 0} \rightarrow \boxed{x \times 0 < x \times x} \qquad [a := x, b := 0]$$

$$\boxed{0 < x} \wedge \boxed{0 < x} \rightarrow \boxed{x \times 0 < x \times x} \qquad [a := 0,\ b := x]$$

$$\boxed{x = 0} \wedge \boxed{x = x} \rightarrow \boxed{x \times x = 0 \times 0} \qquad \begin{matrix}[a := x, b := 0] \\ [c := x,\ d := 0]\end{matrix}$$

$$\boxed{x \times x = 0} \vee \boxed{0 < x \times x}$$

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

$$\forall x, \forall y, \forall z, \left(x = y \land y = z\right) \rightarrow x = z$$

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

$$\forall x, \forall y, x = y \to f(x) = f(y) \qquad \forall x, \forall y, x = y \to P(x) \leftrightarrow P(y)$$

Extends to **multi-argument** functions and relations

$$\forall x_1, \forall y_1, \ldots, x_1 = y_1 \land x_2 = y_2 \land \ldots \to$$

$$f(x_1, x_2, \ldots) = f(y_1, y_2, \ldots) \qquad R(x_1, x_2, \ldots) \leftrightarrow R(y_1, y_2, \ldots)$$

Two Types of Axioms

Construction

Reflection

$$\forall x, x = x$$

Symmetry

$$\forall x, \forall y, x = y \rightarrow y = x$$

Transitivity

$$\forall x, \forall y, \forall z, \left(x = y \wedge y = z\right) \rightarrow x = z$$

Use

Substitution

$$\forall x_1, \forall y_1, \ldots, x_1 = y_1 \wedge x_2 = y_2 \wedge \ldots \rightarrow$$

$$f(x_1, x_2, \ldots) = f(y_1, y_2, \ldots) \qquad R(x_1, x_2, \ldots) \leftrightarrow R(y_1, y_2, \ldots)$$

Example Proof

Symmetry **can be proven** from substitution

$\forall x, \forall y, x = y \to P(x) \leftrightarrow P(y)$ **with** $P(z) := z = x$

$$[\,]\ \forall x, x = x\ ,\ \forall x, \forall y, x = y \to \left(x = x \leftrightarrow y = x\right)\ \vdash\ \forall x, \forall y, x = y \to y = x$$

Example Proof

Symmetry **can be proven** from substitution

$$\overline{\quad A \quad \wedge \quad B \to (A \leftrightarrow C) \quad \to \quad B \to C \quad}$$

$$[x, y] \quad x = x \quad , \quad x = y \to \big(x = x \leftrightarrow y = x\big) \quad \vdash \quad x = y \to y = x$$

$$[x, y] \quad \forall x, x = x \quad , \quad \forall x, \forall y, x = y \to \big(x = x \leftrightarrow y = x\big) \quad \vdash \quad x = y \to y = x$$

$$[\,] \quad \forall x, x = x \quad , \quad \forall x, \forall y, x = y \to \big(x = x \leftrightarrow y = x\big) \quad \vdash \quad \forall x, \forall y, x = y \to y = x$$

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!)

Quantified statements	**Non-quantified statements**
Impossible	O(n log n)

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

Guys check out my book:

Arithmetices principia,
nova methodo exposita

It has first-order axioms.
Also I invented some symbols:

$\wedge$, $\vee$, $\subset$, $\in$

any

axioms

der

axiom system

The Axioms

First:
assume
$0 \leq n$

Axioms for equality (reflexivity, substitution)

Definition of the $x + 1$ operation:

Injectivity

$\forall x, \forall y, x + 1 = y + 1 \rightarrow x = y$

Discrimination

$\forall x, x + 1 \neq 0$

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

First:
assume
$0 \leq n$

Axioms for equality (reflexivity, substitution)

Definition of the $x + 1$ operation

Definition of addition and multiplication

The axiom of **induction**:

$$P(0) \wedge \big(\forall x, P(x) \rightarrow P(x+1)\big) \rightarrow \big(\forall x, P(x)\big)$$

Induction

A ✅ B

$$P(0) \land \big(\forall x, P(x) \to P(x+1)\big) \to \big(\forall x, P(x)\big)$$

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

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

A $P(0) := \exists k, 0 = 2 \times k \lor 0 = 2 \times k + 1$ $\quad k := 0$

$$0 = 2 \times 0 \lor 0 = 2 \times 0 + 1$$

Induction

A ✅ **B**

$$P(0) \land \big(\forall x, P(x) \to P(x+1)\big) \to \big(\forall x, P(x)\big)$$

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

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

B $[\forall n]$ $[n, k]$

$$P(n)$$

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

$$\to$$

$$P(n+1)$$

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

$$\lor$$

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

Induction

A ✅ B ✅ ✅

$$P(0) \land \big(\forall x, P(x) \to P(x+1)\big) \to \big(\forall x, P(x)\big)$$

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

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

B $[n, k]$

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

$$\to$$

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

$$\lor$$

$$n + 1 = 2 \times (k' + 1) \lor n + 1 = 2 \times (k' + 1) + 1$$

Naturals to Integers

Can reason about integers **in terms of** natural numbers

Each integer: difference of a **pair** of natural numbers

$x_+ - x_-$

$$\forall x, \exists y, \exists z, x < y - z \rightarrow x \times y - z < z$$

$\forall x_+, \forall x_-,$

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**

Quantified statements	**Non-quantified statements**
Impossible	Impossible

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

No induction	Robinson Arithmetic	Impossible
No multiplication*	Presberger Arithmetic	O(exp(exp(exp(n))))
No multiplication, quantifiers*	Linear Integer Arithmetic	O(exp(exp(n)))

* 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

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 : Int

Functions

$-Int : Int$

$Int + Int : Int$

$Int \times Int : Int$

$Array[\ Int\ ] : Int$

len$(Array) : Int$

$Array[\ Int := Int\ ] : Array$

Relations

$Int = Int$

$Int < Int$

$Int \in Array$

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”

$$\forall a, \forall x, \forall i, \forall j, i = j \wedge j < \textbf{len}(a) \rightarrow a[i := x][j] = x$$

$$\forall a, \forall x, \forall i, \forall j, i \neq j \wedge j < \textbf{len}(a) \rightarrow a[i := x][j] = a[j]$$

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

Definition of “array contains”

$$\forall a, \forall x, x \in a \leftrightarrow \exists i, i < \textbf{len}(a) \wedge a[i] = x$$

Example

Know $\forall i, \forall j, i < j \to \textbf{left2}[i] < \textbf{left2}[j]^{*}$

$\forall i, \forall j, i < j \to \textbf{right2}[i] < \textbf{right2}[j]^{*}$

$\forall i, \forall j, \textbf{left2}[i] < \textbf{right2}[j]$

Want $\forall i, \forall j, i < j \to (\textbf{left2} + \textbf{right2})[i] < (\textbf{left2} + \textbf{right2})[j]^{*}$

Defining Append

Add axioms to define array append

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

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

$$\forall a, \forall b, \textbf{len}(a + b) = \textbf{len}(a) + \textbf{len}(b)$$

Later, we'll **prove these** from an implementation

DEMO

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 $x = y$

Also called "introduction" and "elimination" rules

Definitions for functions and relations $x + y$

In terms of other, more basic, functions / relations

Induction rules for data structures $x + 1 \mid 0$

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

$$p, q := \neg p \mid p \vee q \mid p \wedge q \mid \forall v, p \mid \exists v, p$$

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

$$[\,]\quad \underbrace{A_1, A_2, \ldots, A_n}_{\text{Axioms}} \quad \vdash \quad \underbrace{p}_{\text{Fact}}$$

Quantifier rules for manipulating facts & axioms

$$\frac{[\Gamma, x]\; A \vdash p}{[\Gamma]\; A \vdash \forall x, p} \qquad \frac{[\Gamma]\; A \vdash p[x := e]}{[\Gamma]\; A \vdash \exists x, p} \qquad \frac{[\Gamma, x]\; A \vdash p}{[\Gamma]\; \exists x, A \vdash p} \qquad \frac{[\Gamma]\; A[x := e] \vdash p}{[\Gamma]\; \forall x, A \vdash p}$$

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

Z3 code for demo

```
;;;; 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