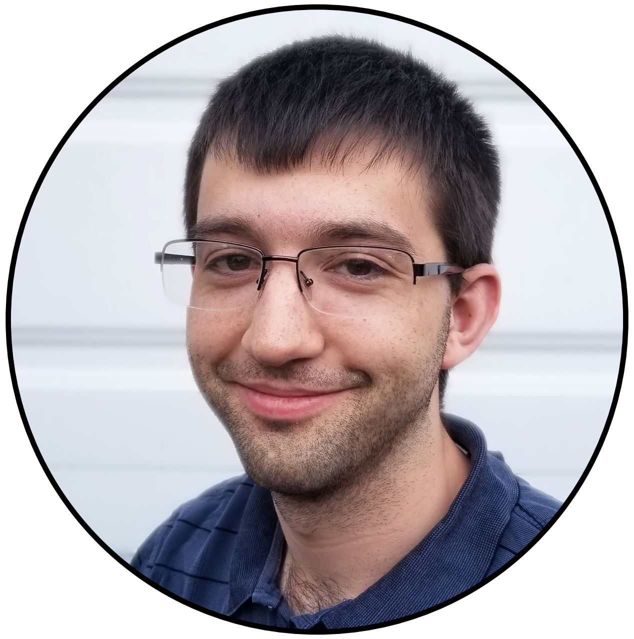
First-order Logic

Specifications section, **Logic** topic, **Lecture 3**



Pavel Panchekha

CS 6110, U of Utah

14 January 2020

Facts about Booleans

$$x, y := \ v \ | \ \neg x \ | \ x \wedge y \ | \ x \vee y$$

Variable

Syntax

That's what you can **say**; what does it **mean?**

Depends on the **values of the variables**

$[\![v]\!] =$ **"v is True"**

$[\![x \wedge y]\!] =$ **both** $[\![x]\!]$ **and** $[\![y]\!]$

$[\![\neg x]\!] =$ **not** $[\![x]\!]$

$[\![x \vee y]\!] =$ **either** $[\![x]\!]$ **or** $[\![y]\!]$ **or both**

Semantics

What use is a Spec?

That's what you can **say**; what does it **mean?**

- Depends on the **values of the variables** :-(

We want to use the spec **before** running a program

- Variables **don't have values** yet!

"**Could it** be true?" "**Must it** be true?" "**If** this, **then** that?"

Satisfiability | Validity | Implication

Evidence

“**Must** $(x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee z) \wedge \neg z$ be false?” **Yes.**

$y \vee z$

z

$\perp$

If $(x \vee y)$ **and** $(\neg x \vee z)$ **then** $(y \vee z)$

$$\frac{x \vee y \qquad \neg x \vee z}{y \vee z}$$

Logical Resolution

DPLL algorithm

Proof by resolution, **presented as a program**

1. Put into conjunctive form
2. Check for **variables all alone**
3. Check for **variables with one polarity**
4. Pick a variable and **try setting** to True
5. If it doesn't work, it **must be** False

Class Progress

Logical reasoning

Program logics

Static analysis

First-order Logic

Decision Procedures

Boolean logic

Syntax

Proof

Theory

First-order Logic

From booleans to **integers**

Addition/multiplication, new kinds of questions

Semantics of **quantifiers**

Alternation, closed terms, and internalization

Generalizing integers to arbitrary data types

The recipe for an arbitrary logic

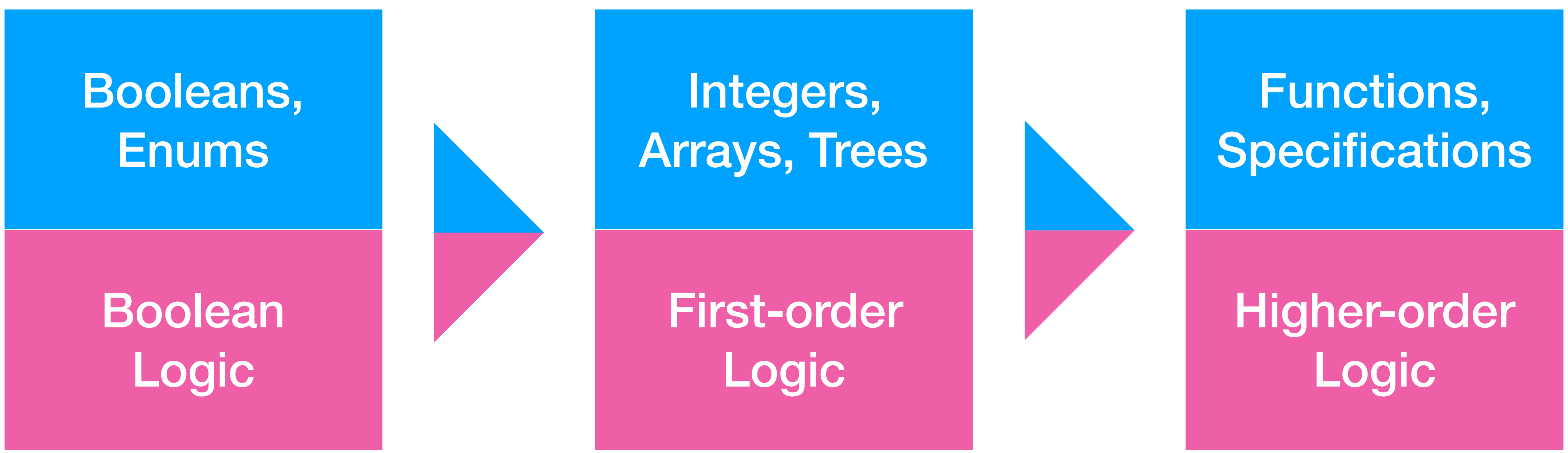
Integer Specifications

A language for arithmetic facts

Specifications

Generally describe **facts about program values**

Today, we go **from boolean to integer values**



Like before, we'll discuss **syntax**, then **semantics**

Boolean Logic Syntax

$$p, q := \boxed{v} \mid \neg p \mid p \wedge q \mid p \vee q$$

Addition, multiplication?

$$x, y := \boxed{v} \mid \boxed{n} \mid -x \mid x + y \mid x \times y$$

Var
i

Constants

Integer Logic Syntax

$$p, q := \neg p \mid p \wedge q \mid p \vee q$$
$$\mid x = y \mid x < y$$

Boolean Expressions

$$x, y := v \mid n \mid -x \mid x + y \mid x \times y$$

Integer Expressions

Equality, comparison?

Examples

Addition is **commutative** (order-independent)

$$x + y = y + x$$

Adding to a comparison on both sides is valid

$$x < y \boxed{\leftrightarrow} x + z < y + z$$

$$p \leftrightarrow q = (p \wedge q) \vee (\neg p \wedge \neg q)$$

Examples

Multiplying a comparison by a positive number is valid

$$\boxed{0 < z \rightarrow} \left(x < y \leftrightarrow x \times z < y \times z\right)$$

Condition on the equivalence

Integers are **either even or odd**

$$\left(n = 2 \times k\right) \vee \left(n = 2 \times k + 1\right)$$

Integer Semantics

In integer logic, **what does a statement mean**?

$\boxed{[\![p]\!] \in \{\textbf{True}, \textbf{False}\}}$

$[\![p \wedge q]\!] =$ **both** $[\![p]\!]$ **and** $[\![q]\!]$

$[\![\neg p]\!] =$ **not** $[\![p]\!]$

$[\![p \vee q]\!] =$ **either** $[\![p]\!]$ **or** $[\![q]\!]$ **or both**

$[\![x = y]\!] = [\![x]\!]$ **equals** $[\![y]\!]$

$[\![x < y]\!] = [\![x]\!]$ **is less than** $[\![y]\!]$

$\boxed{[\![x]\!] \in \{\ldots, -1, 0, 1, \ldots\}}$

$[\![-x]\!] =$ **negative** $[\![x]\!]$

$[\![v]\!] =$ **value of** v

$[\![x \times y]\!] =$ **product of** $[\![x]\!]$ **and** $[\![y]\!]$

$[\![n]\!] =$ **the integer** n

$[\![x + y]\!] =$ **sum of** $[\![x]\!]$ **and** $[\![y]\!]$

Using a Specification

"**Could it** be true?" "**Must it** be true?" "**If** this, **then** that?"

Satisfiability | **Validity** | **Implication**

Next time: what **evidence** can we provide of these?

$$\left(n = 2 \times k\right) \vee \left(n = 2 \times k + 1\right)$$

Satisfiable? **Valid?** **Something else?**

Quantifiers

New questions, internalization, and models

Using a Specification

$$\left(n = 2 \times k\right) \vee \left(n = 2 \times k + 1\right)$$

Satisfiable? **Valid?** **Something else?**

None of the above. Validity on n, satisfiability for k

Must it be the case, for all n

That it **could** be the case, for some k

$$\left(n = 2 \times k\right) \vee \left(n = 2 \times k + 1\right)$$

Complex Questions

Perfect squares are **zero or one** modulo 4

$$\overbrace{\underbrace{n = k \times k}_{\text{For some } k} \implies \underbrace{(n = 4 \times m + 1) \vee (n = 4 \times m)}_{\text{For some } m}}^{\text{For all } n}$$

Solution: include all/some **in the specification itself**

$$p, q := \ldots \mid \boxed{\forall v, p} \mid \boxed{\exists v, p}$$

For all integers v, p is true

For some integer v, p is true

Complex Questions

Perfect squares are **zero or one** modulo 4

For all n

$$\underbrace{\forall n, \left(\underbrace{\exists k, n = k \times k}_{\text{For some } k}\right) \implies \underbrace{\exists m, (n = 4 \times m + 1) \vee (n = 4 \times m)}_{\text{For some } m}}$$

Solution: include all/some **in the specification itself**

$$p, q := \ldots \mid \boxed{\forall v, p} \mid \boxed{\exists v, p}$$

For all integers v, p is true

For some integer v, p is true

Quantifier Semantics

$$p, q := \ldots \mid \forall v, p \mid \exists v, p$$

For all integers v, p is true

For some integer v, p is true

$$\forall x, \underline{\exists y, x < y}$$

For some integer y, $x < y$ is true

What is x?

$$\Gamma = \{x : 3, y : 5, \ \ldots\}$$

$[\![v]\!]_\Gamma =$ **value of** v **in** Γ

$[\![x + y]\!]_\Gamma =$ **sum of** $[\![x]\!]_\Gamma$ **and** $[\![y]\!]_\Gamma$

Quantifier Semantics

$$p, q := \ldots \mid \forall v, p \mid \exists v, p$$

For all integers v, p is true

For some integer v, p is true

$[\![v]\!]_\Gamma =$ **value of** v **in** Γ

$[\![x + y]\!]_\Gamma =$ **sum of** $[\![x]\!]_\Gamma$ **and** $[\![y]\!]_\Gamma$

$[\![\forall v, p]\!]_\Gamma = [\![p]\!]_{\Gamma'}$ **for all** Γ'**, where**

$\Gamma'[x] = \Gamma[x]$ **for all** x **except** v

Semantics of $\forall$

Internalization

$\forall x, p$ → "For all x, p is true" → "p **must be** true"

Quantifiers **internalize** the notion of validity / satisfiability

"p is satisfiable"

$\exists x, \exists y, \ldots, \exists z, p$

"p is valid"

$\forall x, \forall y, \ldots, \forall z, p$

Closed terms (all variables quantified) sufficient

Only one question to ask: is p true?

Examples

$\forall x, \exists y, x < y$

$\exists x, \forall y, x < y$

$\exists x, \exists y, x < y$

$\forall x, \forall y, x < y$

$\forall x, \exists y, x + y = y$

$\exists x, \forall y, x + y = y$

$\forall y, \exists x, x + y = y$

$\exists y, \forall x, x + y = y$

$\exists x, \exists y, x + y = y$

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

$\forall x, \exists y, x = y \times y$

$\exists x, \forall y, x = y \times y$

$\forall y, \exists x, x = y \times y$

$\exists y, \forall x, x = y \times y$

$\exists x, \exists y, x = y \times y$

$\forall x, \forall y, x = y \times y$

Equivalences

$$\neg \forall x, p \leftrightarrow \exists x, \neg p$$

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

Prefix form: all quantifiers at the beginning

$$(\forall x, p) \lor q \to \forall x, p \lor q \qquad (\forall x, p) \land q \to \forall x, p \land q$$

Quantifier alternations are what matter

$$\forall x, \forall y, p \leftrightarrow \forall y, \forall x, p \qquad \forall x, \exists y, p \not\leftrightarrow \exists y, \forall x, p$$

Course Updates

More about the class project

Sample Projects

Class project in **groups of two**. (Result of survey!)

— Symbolic execution for LLVM IR

— Fuzzing existing SMT solvers

— Building a parallel SAT/SMT solver

— Modeling distributed systems in TLA+

Alternative Assignment

Solo **alternative to project**. Released soon.

1. Implement a simple programming language
2. Implement weakest precondition generation
3. Implement static analysis for array bounds
4. Verify a `quicksort` implementation

Beyond Arithmetic

Domains, relations, and theories

Theory of Arrays

How would we write **statements about arrays?**

$$p, q = \ldots \mid x \in a$$

$$x, y = \ldots \mid a[x] \mid \textbf{len}(a)$$

$$a = v \mid a[x := y]$$

New logic for every data type? **No!**

- Syntax
- Semantics
- Equivalences
- Normal forms

Theory of Arrays

First-order Logic

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

$$\mid \; x = y \mid x < y \mid x \in a$$

Relations

Constants

$$x, y := u \mid n \mid -x \mid x + y \mid x \times y \mid a[x] \mid \textbf{len}(a)$$

Functions

$$a := v \mid a[x := y]$$

Sorts

Theory of Arrays

Theory: a set of *sorts*, *constants*, *functions*, and *relations*

Theories are **like programs**, the logic **like an OS**

Sorts

Int

Array

Constants

n : Int

Functions

$-$*Int* : *Int*

Int + *Int* : *Int*

Int × *Int* : *Int*

Array[*Int*] : *Int*

len(*Array*) : *Int*

Array[*Int* := *Int*] : *Int*

Relations

Int = *Int*

Int < *Int*

Int ∈ *Array*

Separate the logic from the data and operations

Syntax

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

For each **sort** T

$$e_T := v_T$$

For each **constant** $c : T$

$$e_T := c$$

For each **function** $f(T_1, T_2, \ldots) : T$

$$e_T := f(e_{T_1}, e_{T_1}, \ldots)$$

For each **relation** $R(T_1, T_2, \ldots)$

$$p := R(e_{T_1}, e_{T_1}, \ldots)$$

Semantics

For each **sort** T | A value set **T** (or "Domain")
$[\![e_T]\!] \in \mathbf{T}$

For each **constant** $c : T$ | A value $\mathbf{c} \in \mathbf{T}$
$[\![c]\!] = \mathbf{c}$

For each **function** $f(T_1, T_2, \ldots) : T$ | A function $\mathbf{f} : \mathbf{T_1}, \mathbf{T_2}, \ldots \rightarrow \mathbf{T}$
$[\![f(a, b, \ldots)]\!] = \mathbf{f}([\![a]\!], [\![b]\!], \ldots)$

For each **relation** $R(T_1, T_2, \ldots)$ | A relation $\mathbf{R} : \mathbf{T_1}, \mathbf{T_2}, \ldots \rightarrow \mathbf{Bool}$
$[\![R(a, b, \ldots)]\!] = \mathbf{R}([\![a]\!], [\![b]\!], \ldots)$

The Recipe

First-order logic

Sorts, constants, functions, relations

\+ **Semantics**

Theory

Examples

	Graphs	Binary search trees
Sorts	Node, Edge	Tree, Value
Constants	None	Empty : *Tree*
Functions	*Edge*.from : *Node* *Edge*.to : *Node*	Node(*Value*, *Tree*, *Tree*) : *Tree* *Tree*.left : *Tree* *Tree*.right : *Tree*
Relations	*Node* = *Node* *Edge* = *Edge*	*Value* < *Value* *Value* ∈ *Tree*

Next class:

First-order Proofs

To do:

- ☐ Course feedback
- ☐ Reading in textbook
- ☐ Assignment 1

First-order Logic

From booleans to **integers**

Addition/multiplication, new kinds of questions

Semantics of **quantifiers**

Alternation, closed terms, and internalization

Generalizing integers to arbitrary data types

The recipe for an arbitrary logic

EVIDENCE

NO TABLES

FINITE PROOFS

Math as Game

Proof Rules

COMPLETENESS

THE MAP

VS

THE TERRITORY

Next class:

First-order Proofs

To do:

- ☐ Course feedback
- ☐ Reading in textbook
- ☐ Assignment 1