Propositions as Types

Static Analysis section, **Lecture 24**

Pavel Panchekha

CS 6110, U of Utah

7 April 2020

What's in an Analysis?

Ingredients you need to define a **task-specific** analysis:

— A set of **conditions** for variables and states

Each condition c corresponds to a property $P_c(x)$

— A **merge function** for conditions

If $c = \text{merge}(c_1, c_2)$, then $\forall x, P_{c_1}(x) \vee P_{c_2}(x) \rightarrow P_c(x)$

Constants, $f(x, y)$, ...

— **Transfer functions** for some functions

If $c = \text{trans}_f(c_1)$, then $\forall x, P_{c_1}(x) \rightarrow P_c(f(x))$

Flow-sensitivity

Sometimes, **if statements** provide additional info:

```
if i >= 0:
  return a[i]
else:
  return a[len(a) - i]
```

$\text{trans}_0() = nn$

$\text{refi}_{\geq}(?, nn) = (nn, nn)$

$i : nn$

Add **refinement functions** for the analysis:

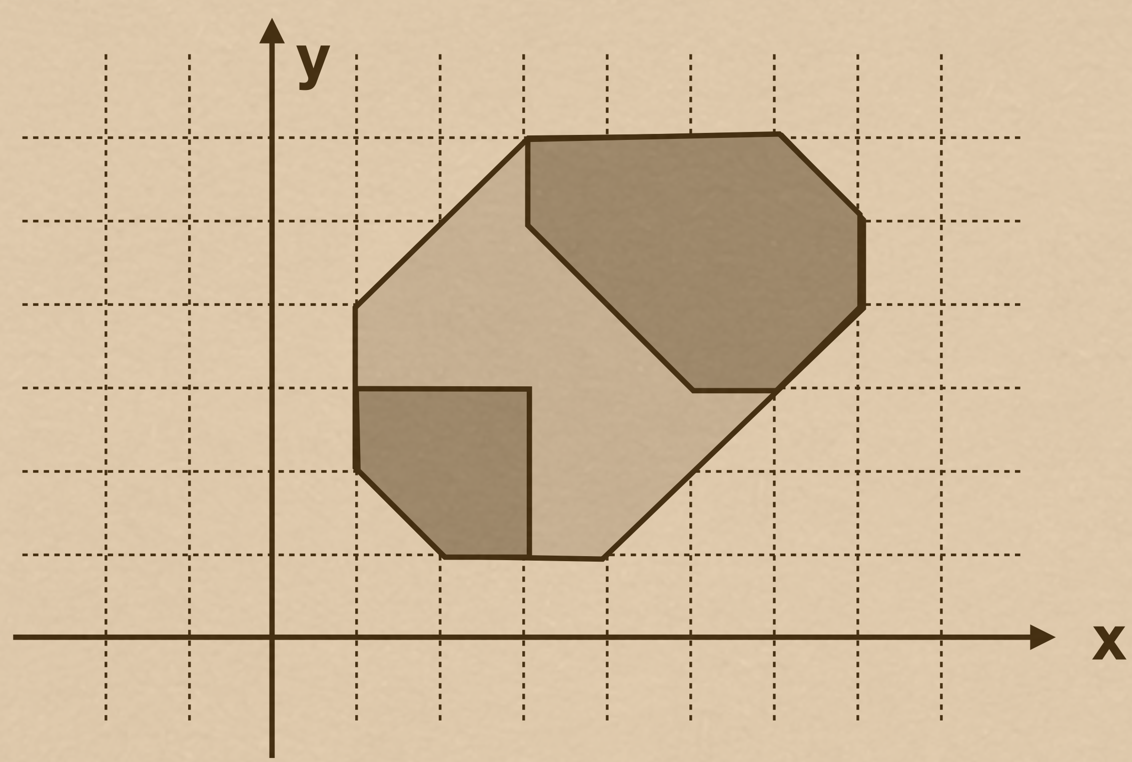
If $(c_1', c_2') = \text{refi}_R(c_1, c_2)$, then

$$\forall x, \forall y, P_{c_1}(x) \wedge P_{c_2}(y) \wedge T(x, y) \rightarrow P_{c_1'}(x) \wedge P_{c_2'}(y)$$

Inequality analysis

Octagon domain: x, y, $x + y$, and $x - y$ ranges

Octagon union handles the four ranges independently



When $x : (a, b)$ and $y : (c, d)$, bounds on $x \pm y$

Class Progress

Logical reasoning	Program logics	Static analysis
Abstract Interpretation		Dependent Types
Propositions as Types	Inductive Types	Type Dependency

Propositions as Types

A **formal language** for writing down proofs

Introduction and elimination rules

Mixing proofs and programs

Computations that return values and proofs

Type theory as a proof language

Assigning terms to proof rules, types to terms

Proof Rules

Introduction and Elimination Rules

Proof rules

Remember these **horizontal line** things?

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

These are called **proof rules**:

$$\frac{\textbf{First prove this}}{\textbf{To prove this}} \Leftrightarrow \frac{\textbf{Assuming this}}{\textbf{Derive this}}$$

Proof rules

Proof rules for proving **and**:

$$\frac{a \qquad\qquad b}{a \wedge b}$$

Proof rules for proving **or**:

$$\frac{a}{a \vee b} \qquad\qquad \frac{b}{a \vee b}$$

Proof Contexts

Theorems often involve **hypothetical reasoning**

"**Let** x **be** a positive integer. Then …"

Assumptions you've made

$$\frac{\boxed{\Gamma} \vdash a}{\boxed{\Gamma} \vdash a \vee b}$$

Implications allow you to describe hypotheticals

Hypothesis to assumption

$$\frac{\boxed{\Gamma, a} \vdash b}{\Gamma \vdash a \to b}$$

Proof Contexts

Assumptions include **variable bindings**

Set of values x is drawn from

$$\frac{\Gamma, \boxed{x : \textbf{int}} \vdash P[x]}{\Gamma \vdash \forall x : \textbf{int}, P[x]}$$

Note similarity to **implication**

We'll come back to this…

$$\frac{\Gamma, \boxed{a} \vdash b}{\Gamma \vdash \boxed{a} \to b} \qquad \frac{\Gamma, \boxed{x : \textbf{int}} \vdash P[x]}{\Gamma \vdash \forall \boxed{x : \textbf{int}}, P[x]}$$

Proof Contexts

Variable binding itself is **hypothetical**

$$a : \textbf{int}, b : \textbf{int} \vdash a + b : \textbf{int}$$

$$a : \textbf{float}, b : \textbf{float} \vdash a + b : \textbf{float}$$

Variable binding itself is **hypothetical**

$$\frac{\Gamma \vdash P[e] \qquad \Gamma \vdash e : \textbf{int}}{\Gamma \vdash \exists x : \textbf{int}, P[x]}$$

Elimination Rules

So far rules had logical connective **below the line**

Called **introduction rules**, they “introduce” a connective

$$\frac{\textbf{First prove this}}{\textbf{To prove this}} \quad \longleftrightarrow \quad \frac{\textbf{Assuming this}}{\textbf{Derive this}}$$

Need opposite rules for **handling assumptions**

The opposite of “introduction” is “elimination” rules

Proof rules

Introduction and elimination rules for **and**:

$$\frac{a \qquad b}{a \wedge b}$$

Introduction

$$\frac{a \wedge b}{a} \qquad \frac{a \wedge b}{b}$$

Elimination

These are **basically opposites**!

More next time…

No "free information" from introducing intermediate steps

More Elimination

Elimination rule for **implication** (“modus ponens”):

$$\frac{a \to b \qquad\qquad a}{b}$$

Elimination rule for **or** (“case analysis”):

$$\frac{a \to c \qquad\qquad a \lor b \qquad\qquad b \to c}{c}$$

Exercise

Write **introduction** and **elimination** rules for negation

Remember: $\neg p \leftrightarrow (p \rightarrow \bot)$

Proof Terms

Program : Type :: Proof : Proposition

Recording Proofs

A valid proof is a series of **proof rules**:

We'd like to **record which rules** were used

Record proof of b

$$\frac{\Gamma, a \vdash b}{\Gamma \vdash a \rightarrow b}$$

Record proof rule

Record assumption

Need to design a **concise language** for recording this

Recording Proofs

Record proof rule

$$\frac{\Gamma, a \vdash b}{\Gamma \vdash a \to b}$$

Recording Proofs

Record proof of b

Record proof rule

$$\frac{\Gamma, a \vdash b}{\Gamma \vdash ii : a \rightarrow b}$$

Recording Proofs

Record proof of b

$$\frac{\Gamma, a \vdash e : b}{\Gamma \vdash ii(e) : a \to b}$$

Record assumption

Recording Proofs

$$\frac{\Gamma, x : a \vdash e : b}{\Gamma \vdash ii(x : a, e) : a \rightarrow b}$$

Change sy

Record assumption

Recording Proofs

$$\frac{\Gamma, x : a \vdash e : b}{\Gamma \vdash (\lambda x : a, e) : a \to b}$$

Implication

What if b mentions x?

$$\frac{\Gamma, x : a \vdash e : b}{\Gamma \vdash (\lambda x : a, e) : a \to b}$$

$$\frac{\Gamma \vdash f : a \to b \qquad \Gamma \vdash e : a}{\Gamma \vdash (f\,e) : b}$$

Generalized Implication

What if b mentions x?

$$\frac{\Gamma, x : a \vdash e : b}{\Gamma \vdash (\lambda x : a, e) : \forall x : a, b}$$

$$\frac{\Gamma \vdash f : \forall x : a, b \qquad\qquad \Gamma \vdash e : a}{\Gamma \vdash (f\, e) : b}$$

And Terms

Proofs of “and” form **structures**:

$$\frac{p_1 : a \qquad\qquad p_2 : b}{(p_1, p_2) : a \wedge b}$$

$$\frac{p \,:\, a \wedge b}{\textbf{fst}\, p \,:\, a} \qquad\qquad \frac{q \,:\, a \wedge b}{\textbf{snd}\, p \,:\, b}$$

Or terms

Proofs of "or" form **unions**:

$$\frac{p : a}{\textbf{lft}\, p : a \lor b} \qquad \frac{p : b}{\textbf{rgt}\, p : a \lor b}$$

$$\frac{f_a : a \to c \qquad p : a \lor b \qquad f_b : b \to c}{\textbf{cases}(p, f_a, f_b) : c}$$

Propositions as Types

Proofs as Programs

Propositions as Types

Proofs are composed of simple **proof rules**

- For each logical operation, **introduction and elimination** rules

Proof terms record which proof rules are used

- Special syntax for each introduction and elimination rule

Proof terms **look like programs**

- Proofs built using functions, structures, and unions

Propositions as Types

Proofs as programs

Propositions as types

Mixing the Two

If proofs are programs and propositions are types…

1. One language for proofs **and** programs

2. Syntax **reused** for programs and proofs

3. Data structures **mix** programs and proofs

4. Proof or program? **You decide!**

Next class:

Inductive Types

To do:

- ☐ Course feedback
- ☐ Class projects

Propositions as Types

A **formal language** for writing down proofs

Introduction and elimination rules

Mixing proofs and programs

Computations that return values and proofs

Type theory as a proof language

Assigning terms to proof rules, types to terms

COMPUTATIONS

PROOFS

ARE

PROGRAMS TOO

INDUCTION

PROOF

BY

RECURSION

TERMINATION

TRICKS AND TECHNIQUE

Next class:

Inductive Types

To do:

- ☐ Course feedback
- ☐ Class projects