Proof Theory and Its Role in Programming Language Research

Frank Pfenning
Carnegie Mellon University

How Do We Write Correct Programs

- We rarely do, but …
- In practice, programming and informal reasoning go hand in hand
  - Operational: how does the program execute
  - Logical: what does it accomplish
- Decompose into parts (e.g., functions, modules) so we can reason locally

Coherence

- Operational and logical views should be coherent
- And both should be as simple as possible
- Composed of parts we can reason about separately as much as possible
  - Not just for programs, but for the language itself
- Logic is inevitable — why wait?

Codesign of Computation and Logic

- Fortunately, logic is computational
- Key: creating a mutual fit — requires considerable ingenuity, persistence, luck
  - Runtime code generation and ??
  - Partial evaluation and ??
  - Dead code elimination and ??
  - Distributed computation and ??
  - Message-passing concurrency and ??
  - ?? and lax logic
  - ?? and temporal logic
  - ?? and epistemic logic
  - ?? and ordered logic

Key Ingredients

- Judgments, leading to propositions
- Basic style of proof system
  - Natural deduction
  - Sequent calculus
  - Axiomatic proof system
  - Binary entailment
- Proof reduction and equality

Example: Hypothetical Judgments

- Basic judgment: A true, for a proposition A
- Hypothetical judgment $\underbrace{A_1\,true, \ldots, A_n\,true}_{\Gamma} \vdash A\,true$
- Defined via substitution property (not rule)

$$\frac{\Gamma \vdash A\ true \quad \Gamma, A\,true \vdash C\ true}{\Gamma \vdash C\ true}\ \mathsf{subst}$$

- Which entails hypothesis rule

$$\frac{}{\Gamma, A\,true \vdash A\ true}\ \mathsf{hyp}$$

With Proof Terms

- Basic judgment: M : A
- Hypothetical judgment = typing judgment

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Gamma} \vdash M : A$$

- Defined via substitution property (dashed line), which entails the hypothesis rule

$$\frac{\Gamma \vdash M : A \quad \Gamma, x{:}A \vdash N : C}{\Gamma \vdash [M/x]N : C}\ \text{subst} \qquad \frac{}{\Gamma, x{:}A \vdash x : A}\ \text{hyp}$$

Internalize Hypothetical Judgment

- Form a proposition whose definition (via an introduction rule) reflects the judgment

$$\frac{\Gamma, A\, true \vdash B\ true}{\Gamma \vdash A \supset B\ true} \supset I$$

- Use the definition of the judgment, to determine the elimination rule

$$\frac{\Gamma \vdash A \supset B\ true \quad \Gamma \vdash A\ true}{\Gamma \vdash B\ true} \supset E$$

Terms Construct and Apply Functions

- Logical rules become familiar typing rules

$$\frac{\Gamma, x{:}A \vdash N : B}{\Gamma \vdash \lambda x.\, N : A \supset B} \supset I \qquad \frac{\Gamma \vdash N : A \supset B \quad \Gamma \vdash M : A}{\Gamma \vdash N\, M : B} \supset E$$

- Introduction rules construct terms
- Elimination rules destruct term
- Computation arises when a destructor is applied to a constructor

Harmony in Natural Deduction

- Introduction rules construct proofs that verify
- Elimination rules construct proof that use
- Harmony between intro and elim rules
  - Any introduction of A followed an elimination of A can be reduced (local reduction)
  - Any proposition A can be proved by an introduction (local expansion)

Proof Reduction is Computation

- On proofs

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \supset B}\supset I \qquad \begin{array}{c}\mathcal{E}\\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\supset E \quad \Longrightarrow \quad \begin{array}{c}\text{subst. } \mathcal{E} \text{ in } \mathcal{D}\\ \Gamma \vdash B\end{array}$$

- On proof terms

$$\dfrac{\dfrac{\Gamma, x{:}A \vdash N : B}{\Gamma \vdash (\lambda x.\, N) : A \supset B}\supset I \qquad \Gamma \vdash M : A}{\Gamma \vdash (\lambda x.\, N)\, M : B}\supset E \quad \Longrightarrow \Gamma \vdash [M/x]N : B$$

Example: Runtime Code Generation

- Key computational idea: we have a quoted source expression available at runtime
- Distinguish
  - Ordinary variables, bound to values
  - Expression variables, bound to source code
- Need to quote and evaluate expressions
  - In a logically correct way

Categorical Judgment

- Judgment form, with variables

$$\underbrace{u_1{:}B_1, \ldots, u_k{:}B_k}_{\text{expression variables}} \;;\; \underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\text{value variables}} \vdash M : A$$

- We can only substitute an expression without reference to value vars for an expression var

$$\frac{\Delta \,;\, \bullet \vdash M : A \quad \Delta, u{:}A \,;\, \Gamma \vdash N : C}{\Delta \,;\, \Gamma \vdash [M/u]N : C}\ \text{esubst}$$

Quotation Continued

- We also have a new hypothesis rule

$$\frac{}{\Delta, u{:}A \,;\, \Gamma \vdash u : A}\ \mathsf{ehyp}$$

- We would like to internalize "A stands for a source expression" as a proposition

Internalizing a Categorical Judgment

- Judgment u:A means A is valid

$$\frac{\Delta \,;\, \bullet \vdash M : A}{\Delta \,;\, \Gamma \vdash \mathsf{quote}\ M : \Box A}\ \Box I$$

$$\frac{\Delta \,;\, \Gamma \vdash M : \Box A \quad \Delta, u : A \,;\, \Gamma \vdash N : C}{\Delta \,;\, \Gamma \vdash (\mathsf{let}\,\mathsf{quote}\, u = M\ \mathsf{in}\ N) : C}\ \Box E$$

- One can check harmony

$$(\mathsf{let}\,\mathsf{quote}\, u = \mathsf{quote}\, M\ \mathsf{in}\ N) \Longrightarrow [M/u]N$$

Which Logic is This?

- Axiomatically, we find

$$\vdash \Box(A \supset B) \supset (\Box A \supset \Box B)$$
$$\vdash \Box A \supset \Box\Box A$$
$$\vdash \Box A \supset A$$

$$\frac{\vdash A}{\vdash \Box A}\ \text{nec}$$

- This defines the intuitionistic modal logic S4
- Conservatively extends intuitionistic logic
- We can have a type theory with quote/eval

Validity and Necessity

- Expression variables correspond to assumptions of validity (u:A ⟺ A valid)
- The box modality internalizes this as a proposition (A valid ⟺ ☐A true)
- Judgmentally, we only need hypothetical and categorical judgments
  - Natural deduction and harmony do the rest
  - Generally, very little "new" is needed

Codesign Revisited

- Runtime code generation and IS4 (A valid)
- Partial evaluation and temporal logic (A @ t)
- Dead code elimination and modal logic IT (A irr)
- Distributed computation and IS5 (A @ w)
- Concurrency and (intuitionistic) linear logic (linear hypothetical judgment)
- Generic effects and lax logic (A lax)
- ?? and epistemic logic (K knows A)
- ?? and ordered logic (ordered hyp. Judgment)

Summary

- Codesign of programming language and its logic can be powerful
  - You'll know when it is right
  - But it is hard
- There are many parameters
  - Style of system (ND, SEQ, HIL, …)
  - Judgments (hypothetical, categorical, linear, …)
  - Relating proof reduction to computation
  - Equality, for a full type theory

Some Advice

- Focus on what you can express, not what you can't
- Measure success by the constructs omitted, not those included
- Design, program and reason, iterate
- Syntax is important
- Semantics is even more important, both operational and logical
- Know when to give up