Purifying Natural Deduction Using Sequent Calculus

Aaron Stump

Computational Logic Center
CS, The University of Iowa

Funding from NSF CAREER.

Verified Programming

Thesis

The ability to state and prove properties of code is the crucial missing technology in the evolution of software.

- Stronger guarantees => less monitoring => higher performance.
- Ability to trust software opens up new applications.
- Confirmed quality helps open source, app stores, etc.
- Verification is a tool we don't have.

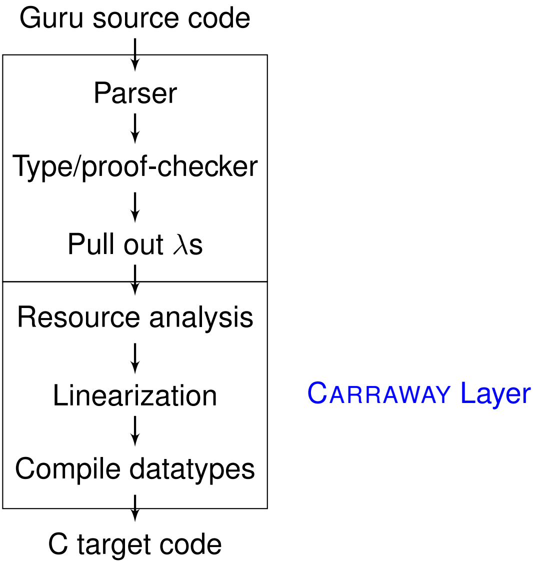
The GURU Verified Programming Language (VPL)

- Functional language
- Dependently typed programs
- General recursion
- Notation for theorems, proofs about programs
- Unaliased mutable state
- Resource management layer
- Type/Proof-checker, compiler to C
- No concurrency
- Aliasing for mutable state in progress

Basic GURU Design

- Terms : Types.
- Proofs : Formulas.
- "Full-spectrum" dependency.
  - Types can contain arbitrary terms (`<list A n>`).
  - Type checking decidable.
  - Explicit casts with proofs of `{T = T'}`.
- Proofs and types can appear in terms.
  - computationally irrelevant.
  - erased by compilation, definitional equality.

The GURU Compiler



Resource Management in GURU

- Resources: program data, I/O channels, mutable arrays.
- Resource typing side-by-side with data typing.
- Management policies definable.
- Based on fundamental idea of a resource:
  1. A resource can only be used by one entity at a time.
  2. A resource can be temporarily decomposed into subresources.
- Statically ensure all resources “consumed” exactly once.

Subresources

- "Goblet of Fire" as subresource of Harry Potter boxed set.
- Sublist l′ as a subresource of (cons x l′).
- Subresource relationship based on type <R x>:
  - x:R – x has resource type R.
  - y:<R′ x> – y has resource type R', and is a subresource of x.
- Cannot consume x until all subresources have been consumed.

Example: Reference-Counted Data

- GURU uses reference counting for inductive data.
- Primitive `(inc x)` creates new view of `x`.
- `(dec x)` consumes a view of `x`.
- `owned` resource type for loaned reference.

```
match l with              % suppose l:<owned x>
  nil => ...
| cons x l' =>            % then l' : <owned l>
```

- Must drop `l'` before consuming `l`.
- Can increment `l'` to get new view.
- Sometimes must collapse chains of ownership:

```
@ l' : <owned x>
```

Meta-Theoretic Concerns

- To implement a VPL: go from proof theory to compilers.
- "Practical" proof theory lacking.
- Problems with disjunctions ($\phi \vee \phi'$) and existentials ($\exists x.\phi$).
- Rest of the talk: the problems, and progress towards a solution.

Practical Proof Theory

- How to prove your logic is consistent?
- Basic strategy:
  1. Identify subset of proofs which obviously are ok.
  2. Define rewrite rules to transform any proof to one in the ok form.
  3. Prove rules are (strongly or weakly) normalizing.
- By Curry-Howard isomorphism:
  - Proofs are λ-terms.
  - Proof normalization is β-reduction.
- Reducibility proofs are powerful, elegant.
- But do not work well with disjunctions, existentials.

Reducibility for Conjunction

Proof terms $p ::= (p_1, p_2) \mid p.1 \mid p.2$

$$\frac{\Gamma \vdash p_1 : \phi_1 \quad \Gamma \vdash p_2 : \phi_2}{\Gamma \vdash (p_1, p_2) : \phi_1 \wedge \phi_2} \wedge\text{I}$$

$$\frac{\Gamma \vdash p : \phi_1 \wedge \phi_2 \quad i \in \{1, 2\}}{\Gamma \vdash p.i : \phi_i} \wedge\text{E}$$

Reducibility is “hereditary normalization”, defined by eliminations.

- Red_ϕ is set of reducible terms of type ϕ.
- $p \in Red_b \;\Leftrightarrow\; SN(p)$, for base types b.
- $p \in Red_{\phi_1 \wedge \phi_2} \;\Leftrightarrow\; p.1 \in Red_{\phi_1}$ and $p.2 \in Red_{\phi_2}$.
- $p \in Red_{\phi_1 \rightarrow \phi_2} \;\Leftrightarrow\;$ forall $p' \in Red_{\phi_1}, (p\ p') \in Red_{\phi_2}$

What Goes Wrong with Disjunction

Proof terms $p ::= \langle 1, p \rangle \mid \langle 2, p \rangle \mid case(p)(x.p_1, x.p_2)$

$$\frac{\Gamma \vdash p : \phi_i \quad i \in \{1, 2\}}{\Gamma \vdash \langle i, p \rangle : \phi_1 \wedge \phi_2} \ \vee\mathrm{I}$$

$$\frac{\Gamma \vdash p : \phi_1 \vee \phi_2 \quad \Gamma, x : \phi_1 \vdash p_1 : \psi \quad \Gamma, x : \phi_2 \vdash p_2 : \psi}{\Gamma \vdash case(p)(x.p_1, x.p_2) : \psi} \ \vee\mathrm{E}$$

Attempt to define reducibility fails:

$$p \in Red_{\phi_1 \vee \phi_2} \ \Leftrightarrow \ \text{forall } \psi, \ p_1, p_2 \in Red_\psi, case(p)(x.p_1, x.p_2) \in Red_\psi$$

Not legal to appeal to Red_ψ.

A Way Forward

- Problem with $\vee$E:
  - to use $p : \phi$, need $p' : \psi$, where ψ unrelated to ϕ.
  - breaks definition of reducibility.
- But compare sequent calculus rules:

$$\frac{\Gamma, \phi_1 \vdash \psi \quad \Gamma, \phi_2 \vdash \psi}{\Gamma, \phi_1 \vee \phi_2 \vdash \psi}\ \mathsf{L}\vee \qquad \frac{\Gamma, \phi_1, \phi_2 \vdash \psi}{\Gamma, \phi_1 \wedge \phi_2 \vdash \psi}\ \mathsf{L}\wedge$$

- Term assignment for sequent calculus is strange.

$$\frac{\Gamma, y : \phi_1, z : \phi_2 \vdash p : \psi}{\Gamma, x : \phi_1 \wedge \phi_2 \vdash [x.1/y, x.2/z]p : \psi}\ \mathsf{L}\wedge$$

- Limited by old view of “natural” deduction.

A Direct Term Assignment

- Left rules correspond to eliminations.
- Why insist that the context Γ holds just variables?
- Proposal:
  - Assign terms to sequent calculus directly.
  - Devise new terms for $\vee$E, $\exists$E.
  - Allow Γ to hold terms.

Elimination Rules

$$\frac{\Gamma, p.1 : \phi_1, p.2 : \phi_2 \vdash p' : \psi}{\Gamma, p : \phi_1 \wedge \phi_2 \vdash p' : \psi}\ \mathsf{L}\wedge$$

$$\frac{\begin{array}{c}\Gamma, p.(1) : \phi_1 \vdash p_1 : \psi \\ \Gamma, p.(2) : \phi_2 \vdash p_2 : \psi\end{array}}{\Gamma, p : \phi_1 \vee \phi_2 \vdash p_1 \,||\, p_2 : \psi}\ \mathsf{L}\vee$$

$$\frac{\Gamma, (p\ a) : [a/x]\phi \vdash p' : \psi}{\Gamma, p : \forall x.\phi \vdash p' : \psi}\ \mathsf{L}\forall$$

$$\frac{\Gamma, p!x : \phi \vdash p' : \psi \quad x \notin FV(\Gamma, \psi)}{\Gamma, p : \exists x.\phi \vdash \nu x.p' : \psi}\ \mathsf{L}\exists$$

$$\frac{}{p : \phi \vdash p : \phi}\ \mathsf{Ax}$$

$$\frac{\Gamma \vdash p_2 : \phi_2 \quad \Gamma, (p_1\ p_2) : \phi_1 \vdash p' : \psi}{\Gamma, p_1 : \phi_2 \to \phi_1 \vdash p' : \psi}\ \mathsf{L}\!\to$$

$$\frac{\Gamma \vdash p' : \psi}{\Gamma, p : \phi \vdash [p]p' : \psi}\ \mathsf{LW}$$

$$\frac{\Gamma, p : \phi, p : \phi \vdash p' : \psi}{\Gamma, p : \phi \vdash p' : \psi}\ \mathsf{LC}$$

Reduction

- We have separated logical terms ($t.(i)$) from structural ($t_1 \,||\, t_2$).
- Logical terms have β-reductions:

$$\begin{array}{l} (t_1, t_2).i \;\rightsquigarrow\; t.i \\ \langle i, t\rangle.(i) \;\rightsquigarrow\; t \\ \langle i, t\rangle.(3-i) \;\rightsquigarrow\; \mathit{abort} \end{array}$$

- Structural terms have commuting conversions:

$$\begin{array}{l} (t_1 \,||\, t_2).i \;\rightsquigarrow\; (t_1.i) \,||\, (t_2.i) \\ \mathit{abort} \,||\, t \;\rightsquigarrow\; t \end{array}$$

- Simple unsound typing rules suffice for reducibility.

$$\frac{\Gamma \vdash p : \phi_1 \vee \phi_2}{\Gamma \vdash p.i : \phi_i} \vee\mathrm{E}$$

Towards Pure Natural Deduction

- Next step: define sound natural deduction rules.

$$\begin{array}{rcl} J & ::= & \Gamma \vdash \Delta \ \mid \ J \,||\, J \\ \Delta & ::= & t_1 : \phi_1, \ldots, t_n : \phi_n \end{array}$$

- Prove type preservation.
- Prove confluence.
- Final result: Pure Natural Deduction.
  - All rules are either direct logical rules or structural.
  - Consistency proved by reducibility.
  - Decidable equational theory, including commuting conversions.
  - Practical proof theory ready to use for VPL.

www.guru-lang.org