Focusing on Binding and Computation

Dan Licata

Joint work with Noam Zeilberger and Robert Harper

Carnegie Mellon University

Programming with Proofs

- Represent syntax, judgements, and proofs
- Reason about them via computation

Programming with Proofs

- Represent syntax, judgements, and proofs
  - ▷ Binding and scope!
- Reason about them via computation
  - ▷ Structural induction modulo α-equivalence

Programming with Proofs

- Represent syntax, judgements, and proofs
  - ▷ Binding and scope!
- Reason about them via computation
  - ▷ Structural induction modulo α-equivalence

Logical frameworks: abstractions facilitating these tasks

Programming with Proofs

- Represent syntax, judgements, and proofs
  - ▷ Binding and scope!
- Reason about them via computation
  - ▷ Structural induction modulo α-equivalence

Logical frameworks: abstractions facilitating these tasks

What theory of inference rules?

Derivability

$$\frac{A \text{ true} \vdash B \text{ true}}{(A \supset B) \text{ true}}$$

- $J_1 \vdash J_2$: derive J_2, using a new axiom concluding J_1
- Does **not** circumscribe J_1
- Structural properties:
  substitution, weakening, exchange, contraction

Admissibility

$$\frac{P(0)\ \text{true} \quad P(1)\ \text{true} \quad \ldots}{\forall x : \mathbb{N}.P(x)\ \text{true}} \qquad \text{i.e.} \qquad \frac{n : \mathbb{N} \models P(n)\ \text{true}}{\forall x : \mathbb{N}.P(x)\ \text{true}}$$

- $J_1 \models J_2$: **if** J_1 is derivable **then** J_2 is derivable (implication in metalogic)
- **Does** circumscribe J_1
  e.g. by distinguishing all possible cases on $n : \mathbb{N}$

Admissibility

Side conditions:

$$\frac{l \notin M}{(M, l) \hookrightarrow \mathsf{error}} \quad \text{i.e.} \quad \frac{(l \in M) \models \bot}{(M, l) \hookrightarrow \mathsf{error}}$$

Iterated inductive definitions:

$$\frac{\mathsf{path}(x, y, n) \qquad (\mathsf{path}(x, y, m) \models m \geq n)}{\mathsf{shortestPath}(x, y, n)}$$

Evidence

1. Evidence for admissibility $J_1 \models J_2$:
   Open-ended: any transformation from J_1 to J_2

   Called computational functions (cf. Coq, NuPRL)

Evidence

1. Evidence for admissibility $J_1 \models J_2$:
   Open-ended: any transformation from J_1 to J_2

   Called computational functions (cf. Coq, NuPRL)

2. Evidence for derivability $J_1 \vdash J_2$:
   - a uniform function: may not analyze J_1
   - application = substitution
   - accounts for syntax with variable binding

   Called representational functions (cf. LF)

Focusing on Binding and Computation

This work:

A single (simply-typed) logical framework supporting both binding and computation.

- Two functions spaces:
  representational arrow $\Rightarrow$ for derivability
  computational arrow $\rightarrow$ for admissibility
- Inference rules can freely mix them

Representational Arrow

Intro: $(\lambda\, u.\ V) : P \Rightarrow A$

- V a value of type A
- u is a scoped datatype constructor for P

Examples of $P \Rightarrow P$:

$$\lambda\, u.\ u$$
$$\lambda\, u.\, \mathsf{c}\ u \qquad \text{(if } \mathsf{c} : (P \Rightarrow P)\text{)}$$

Elim: Pattern matching

$$\begin{array}{lrcl} \mathsf{case}\ (e : P \Rightarrow P)\ \mathsf{of} & \lambda\, u.\ u & \mapsto & e_1 \\ & \mid \lambda\, u.\, \mathsf{c}\ u & \mapsto & e_2 \\ & \vdots & & \end{array}$$

Outline

What?

- Motivating example
- Structural properties

How?

- Polarity of $\Rightarrow$
- Higher-order focusing for intuitionistic logic
- Computational open-endedness of inversion

Outline

What?

- **Motivating example**
- Structural properties

How?

- Polarity of $\Rightarrow$
- Higher-order focusing for intuitionistic logic
- Computational open-endedness of inversion

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$e \quad ::= \quad \mathsf{num}[k]$$
$$\mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$$

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & & \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & & \mid e_1 + e_2
\end{array}
$$

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & | & \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & | & e_1 + e_2 \\
 & | & e_1 * e_2
\end{array}
$$

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & | & \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & | & e_1 + e_2 \\
 & | & e_1 * e_2 \\
 & | & e_1 - e_2
\end{array}
$$

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & | & \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & | & e_1 + e_2 \\
 & | & e_1 * e_2 \\
 & | & e_1 - e_2 \\
 & | & e_1\ \mathsf{div}\ e_2
\end{array}
$$

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & | & \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & | & e_1 + e_2 \\
 & | & e_1 * e_2 \\
 & | & e_1 - e_2 \\
 & | & e_1\ \mathsf{div}\ e_2 \\
 & | & e_1\ \mathsf{mod}\ e_2
\end{array}
$$

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & | & \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & | & e_1 + e_2 \\
 & | & e_1 * e_2 \\
 & | & e_1 - e_2 \\
 & | & e_1\ \mathsf{div}\ e_2 \\
 & | & e_1\ \mathsf{mod}\ e_2 \\
 & | & e_1\ \mathsf{pow}\ e_2
\end{array}
$$

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & | & \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & | & e_1 + e_2 \\
 & | & e_1 * e_2 \\
 & | & e_1 - e_2 \\
 & | & e_1\ \mathsf{div}\ e_2 \\
 & | & e_1\ \mathsf{mod}\ e_2 \\
 & | & e_1\ \mathsf{pow}\ e_2
\end{array}
$$

Suppose we want to treat binops unformly

Example: Arithmetic Expressions

Language of arithmetic expressions:

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
 & | & \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
 & | & e_1 \odot_f e_2
\end{array}
$$

Represent binops generically by

$$f : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$$

Example: Arithmetic Expressions

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
& & \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
& & \mid e_1 \odot_f e_2
\end{array}
$$

Represent in our framework as type ari with constructors:

$$\mathsf{num} : \mathsf{ari} \Leftarrow \mathsf{nat}$$

$$\mathsf{let} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{ari} \Rightarrow \mathsf{ari})$$

$$\mathsf{binop} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}) \Leftarrow \mathsf{ari}$$

Example: Arithmetic Expressions

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
& & \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
& & \mid e_1 \odot_f e_2
\end{array}
$$

Represent in our framework as type ari with constructors:

$$
\begin{array}{l}
\mathsf{num} : \mathsf{ari} \Leftarrow \mathsf{nat} \\
\mathsf{let} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{ari} \Rightarrow \mathsf{ari}) \\
\mathsf{binop} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{nat} \rightarrow \mathsf{nat} \rightarrow \mathsf{nat}) \Leftarrow \mathsf{ari}
\end{array}
$$

Uses representational function for let

Example: Arithmetic Expressions

$$
\begin{array}{lll}
e & ::= & \mathsf{num}[k] \\
& & \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 \\
& & \mid e_1 \odot_f e_2
\end{array}
$$

Represent in our framework as type ari with constructors:

$$\mathsf{num} : \mathsf{ari} \Leftarrow \mathsf{nat}$$

$$\mathsf{let} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{ari} \Rightarrow \mathsf{ari})$$

$$\mathsf{binop} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}) \Leftarrow \mathsf{ari}$$

Uses computational function for binop

Example: Evaluator

$ev : \mathsf{ari} \longrightarrow \mathsf{nat}$

$$
\begin{array}{lll}
ev\ (\mathsf{num}\ p) & \mapsto & p \\
ev\ (\mathsf{binop}\ p_1\ f\ p_2) & \mapsto & f\ (ev\ p_1)\ (ev\ p_2) \\
ev\ (\mathsf{let}\ p_0\ (\lambda\, u.\, p)) & \mapsto & ev\ (apply\ (\lambda\, u.\, p)\ p_0)
\end{array}
$$

Example: Evaluator

$$\mathit{ev} : \mathsf{ari} \longrightarrow \mathsf{nat}$$

$$\begin{aligned}
&\mathit{ev}\ (\mathsf{num}\ p) && \mapsto p \\
&\mathit{ev}\ (\mathsf{binop}\ p_1\ \ f\ \ p_2) && \mapsto f\ (\mathit{ev}\ p_1)\ (\mathit{ev}\ p_2) \\
&\mathit{ev}\ (\mathsf{let}\ p_0\ \ (\lambda\, u.\, p)) && \mapsto \mathit{ev}\ (\mathit{apply}\ (\lambda\, u.\, p)\ p_0)
\end{aligned}$$

apply a representational function by substitution:

$$\mathit{apply} : (P \Rightarrow A) \longrightarrow (P \longrightarrow A)$$

Outline

What?

- Motivating example
- **Structural properties**

How?

- Polarity of $\Rightarrow$
- Higher-order focusing for intuitionistic logic
- Computational open-endedness of inversion

Structural Properties

- Properties of derivability judgement $J_1 \vdash J_2$:

$$\textit{apply} : (P \Rightarrow A) \rightarrow (P \rightarrow A)$$
$$\textit{weaken} : A \rightarrow (P \Rightarrow A)$$

- "Free" in LF: all rules are pure

*May **fail** if rules mix derivability and admissibility!*

Counterexample to Weakening

$$\mathit{weaken} : A \to (P \Rightarrow A)$$

Counterexample:

$$\mathit{plus} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$$

defined by recursion on nat.

Cannot *weaken to* $\mathsf{nat} \Rightarrow \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$*:*
would introduce a new case for plus

Our Solution

$\Rightarrow$ eliminated by pattern-matching:

- No commitment to *apply*, *weaken*
- But structural properties are definable for all LF rules, and in many other cases. E.g.

$$\textit{weaken} : A \rightarrow (P \Rightarrow A)$$

  if P does not occur to the left of computational arrow

- Implement as a datatype-generic program

Outline

What?

- Motivating example
- Structural properties

How?

- **Polarity of** $\Rightarrow$
- Higher-order focusing for intuitionistic logic
- Computational open-endedness of inversion

Intro vs. Elim

Sums $A \oplus B$:

- Introduced by choosing inl or inr
- Eliminated by pattern-matching

Computational functions $A \to B$:

- Introduced by pattern-matching on A
- Eliminated by choosing an A to apply it to

Positive vs. Negative Polarity [Girard '93]

Sums $A \oplus B$ are positive:

- Introduced by choosing inl or inr
- Eliminated by pattern-matching

Computational functions $A \to B$ are negative:

- Introduced by pattern-matching on A
- Eliminated by choosing an A to apply it to

Focus vs. Inversion [Andreoli '92]

Sums $A \oplus B$ are positive:

- Introduced by choosing inl or inr
- Eliminated by pattern-matching

Computational functions $A \to B$ are negative:

- Introduced by pattern-matching on A
- Eliminated by choosing an A to apply it to

Focus vs. Inversion [Andreoli '92]

Sums $A \oplus B$ are positive:

- Introduced by choosing inl or inr
- Eliminated by pattern-matching

Computational functions $A \to B$ are negative:

- Introduced by pattern-matching on A
- Eliminated by choosing an A to apply it to

Focus = make choices

Focus vs. Inversion [Andreoli '92]

Sums $A \oplus B$ are positive:

- Introduced by choosing inl or inr
- Eliminated by pattern-matching

Computational functions $A \to B$ are negative:

- Introduced by pattern-matching on A
- Eliminated by choosing an A to apply it to

Focus vs. Inversion [Andreoli '92]

Sums $A \oplus B$ are positive:

- Introduced by choosing inl or inr
- Eliminated by pattern-matching

Computational functions $A \to B$ are negative:

- Introduced by pattern-matching on A
- Eliminated by choosing an A to apply it to

Inversion = respond to all possible choices

Representational Functions are Positive

- Specified by intro: $\lambda\, u.\ V$
- Eliminated by pattern matching:

$$\mathsf{case}\ (e : P \Rightarrow A)\ \mathsf{of}\ \{(\lambda\, u.\ p)\ \mapsto\ e\}$$

where p is in an extended rule context

Outline

What?

- Motivating example
- Structural properties

How?

- Polarity of $\Rightarrow$
- **Higher-order focusing for intuitionistic logic**
- Computational open-endedness of inversion

Higher-order Focusing

1. Specify a type by its patterns

2. Type-independent focusing framework:
   - Focus phase = choose a pattern
   - Inversion phase = pattern matching

See Zeilberger [APAL] for classical logic
and Zeilberger [POPL08] for positive half of IL

Sequent Calculus Judgements

Type-specific:

- Constructor patterns $\Delta \Vdash p :: C^{+}$
  and destructor patterns $\Delta \Vdash n :: C^{-} > C^{+}$

Focusing framework:

- Positive focus $\Gamma \vdash v^{+} :: C^{+}$
  and inversion $\Gamma \vdash k^{+} : C_0^{+} > C^{+}$
- Negative focus $\Gamma \vdash k^{-} :: C^{-} > C^{+}$
  and inversion $\Gamma \vdash v^{-} : C^{-}$
- Neutral sequents $\Gamma \vdash e : C^{+}$
  and substitutions $\Gamma \vdash \sigma : \Delta$

Sequent Calculus Judgements

Type-specific:

- **Constructor patterns** $\Delta \Vdash p :: C^{+}$
  and destructor patterns $\Delta \Vdash n :: C^{-} > C^{+}$

Focusing framework:

- **Positive focus** $\Gamma \vdash v^{+} :: C^{+}$
  and inversion $\Gamma \vdash k^{+} : C_0^{+} > C^{+}$
- Negative focus $\Gamma \vdash k^{-} :: C^{-} > C^{+}$
  and inversion $\Gamma \vdash v^{-} : C^{-}$
- Neutral sequents $\Gamma \vdash e : C^{+}$
  and substitutions $\Gamma \vdash \sigma : \Delta$

Sequent Calculus Judgements

Judgements relative to inference rule context Ψ:

$$\begin{array}{lll} R & ::= & P \Leftarrow A_1^+ \cdots \Leftarrow A_n^+ \\ \Psi & ::= & \cdot \mid \Psi, u : R \end{array}$$

Natural numbers:

$$\begin{array}{lll} \Psi_{\mathsf{nat}} & = & \mathsf{zero} : \mathsf{nat} \\ & & \mathsf{succ} : \mathsf{nat} \Leftarrow \mathsf{nat} \end{array}$$

Cf. definitional reflection [Schroeder-Heister/Hallnäs]

Sequent Calculus Judgements

Assumptions and conclusions are *contextual*:
Track the free variables of a term in its type

[cf. Contextual Modal Type Theory and FO$\lambda^{\Delta\nabla}$]

$$
\begin{array}{rcl}
\Gamma, \Delta & ::= & \cdot \mid \Delta, x : C^{-} \\
C^{-} & ::= & \langle \Psi \rangle \, A^{-} \\
C^{+} & ::= & \langle \Psi \rangle \, A^{+}
\end{array}
$$

Patterns

Constructor Patterns: $\Delta \Vdash p :: \langle \Psi \rangle\, A^{+}$

$$
\begin{array}{lcl}
A^{+} & ::= & \downarrow A^{-} \mid P \mid R \Rightarrow A^{+} \\
A^{-} & ::= & A^{+} \rightarrow B^{-} \mid \uparrow A^{+}
\end{array}
$$

$$
\frac{}{x : \langle \Psi \rangle\, A^{-} \Vdash x :: \langle \Psi \rangle \downarrow A^{-}}
$$

Constructor Patterns: $\Delta \Vdash p :: \langle \Psi \rangle\, A^{+}$

$$\frac{\begin{array}{l} u : P \Leftarrow A_1^{+} \cdots \Leftarrow A_n^{+} \in \Psi \\ \Delta_1 \Vdash p_1 :: \langle \Psi \rangle\, A_1^{+} \\ \vdots \\ \Delta_n \Vdash p_n :: \langle \Psi \rangle\, A_n^{+} \end{array}}{\Delta_1, \ldots, \Delta_n \Vdash u\ p_1 \ldots p_n :: \langle \Psi \rangle\, P}$$

Constructor Patterns: $\Delta \Vdash p :: \langle \Psi \rangle\, A^+$

$$\frac{\Delta \Vdash p :: \langle \Psi, u : R \rangle\, A^+}{\Delta \Vdash \lambda\, u.\, p :: \langle \Psi \rangle\, R \Rightarrow A^+}$$

- $R \Rightarrow A^+$ binds a scoped datatype constructor
- Can pattern-match through a λ
- “Shocking” type isomorphisms:

$$R \Rightarrow (A^+ \oplus B^+) \cong (R \Rightarrow A^+) \oplus (R \Rightarrow B^+)$$

Focusing Framework

Positive Focus: $\Gamma \vdash v^{+} :: C^{+}$

$$\frac{\Delta \Vdash p :: C^{+} \quad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash p\,[\sigma] :: C^{+}}$$

- Positive value is pattern p with substitution σ
- σ substitutes negative values v^{-}/x for $x : C^{-} \in \Delta$

Positive Inversion: $\Gamma \vdash k^{+} : C^{+} > D^{+}$

$$\frac{\forall(\Delta \Vdash p :: C^{+}).\ \Gamma, \Delta \vdash \phi(p) : D^{+}}{\Gamma \vdash \mathsf{cont}^{+}(\phi) : C^{+} > D^{+}}$$

- Positive continuation is a case-analysis
- Higher-order: specified by meta-level function

$$\phi = \{p \mapsto e, \ldots\}$$

from patterns to expressions

Cut Admissibility

Theorem

1. Positive cut: If $\Gamma \vdash v^+ :: C^+$ and $\Gamma \vdash k^+ : C^+ > D^+$ then $\Gamma \vdash v^+ \bullet k^+ : D^+$

2. Negative cut: If $\Gamma \vdash v^- : C^-$ and $\Gamma \vdash k^- :: C^- > D^+$ then then $\Gamma \vdash v^- \bullet k^- : D^+$

3. Substitution: If $\Gamma, \Delta \vdash \mathcal{J}$ and $\Gamma \vdash \sigma : \Delta$ then $\Gamma \vdash \mathcal{J}\,[\sigma]$

Cut Admissibility

Procedure is independent of connectives
E.g. for positive cut:

$$\frac{\Delta \Vdash p :: C^+ \quad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash p\,[\sigma] :: C^+} \qquad \frac{\forall(\Delta \Vdash p :: C^+).\ \Gamma, \Delta \vdash \phi(p) : D^+}{\Gamma \vdash \mathsf{cont}^+(\phi) : C^+ > D^+}$$

$$(p\,[\sigma]) \bullet \mathsf{cont}^+(\phi) \quad = \quad \phi(p)\,[\sigma]$$

Cut Admissibility

Procedure is independent of connectives
E.g. for positive cut:

$$\frac{\Delta \Vdash p :: C^+ \quad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash p\,[\sigma] :: C^+} \qquad \frac{\forall(\Delta \Vdash p :: C^+).\ \Gamma, \Delta \vdash \phi(p) : D^+}{\Gamma \vdash \mathsf{cont}^+(\phi) : C^+ > D^+}$$

$$(p\,[\sigma]) \bullet \mathsf{cont}^+(\phi) \quad = \quad \phi(p)\,[\sigma]$$

Termination depends on subformula property

Outline

What?

- Motivating example
- Structural properties

How?

- Polarity of $\Rightarrow$
- Higher-order focusing for intuitionistic logic
- **Computational open-endedness of inversion**

Computational Open-endedness

Inversion may have infinitely many cases:

$$\cdot \vdash \mathsf{cont}^{+}(\phi) : \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{ari} > \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{nat}$$

In extension:

- ϕ must give one case for each ari expression, except
- bind variables in Δ for $\rightarrow$ functions from binop

Any method of presenting ϕ is acceptable!

Computational Open-endedness

1. May present ϕ as function in existing proof-assistant, reusing its pattern coverage checker
   - Opportunity for datatype-generic programs

   Agda implementation on the Web!

2. Or design a traditional finitary syntax (future work)

3. Theory accounts for “foreign-function interface” to existing tools

Related Work

Our approach is different than

- LF/Twelf, because we permit computation in data
- FO$\lambda^{\Delta\nabla}$, because $\Rightarrow$ introduces a fresh inference rule, not a fresh individual
- nominal logic, because we don't separate name generation from name binding (therefore no effects)

Related Work

Our approach is different than

- dependent de Bruijn indices, because structural properties are implemented type-generically
- weak HOAS / hybrid approaches, because we represent binding as positive data—can pattern match through $\Rightarrow$

Conclusion

What?

- Simply-typed framework for rules that mix $\Rightarrow$ and $\rightarrow$
  - $\triangleright$ Future work: dependency on data, computation
- Structural properties implemented generically, under certain conditions

How?

- Higher-order focusing
- Contextual hypotheses and conclusions

Thanks for listening!

Higher-order Focusing for Intuitionistic Logic

Polarity and Focusing

	Positive type	Negative type
Intro	Focus	Inversion
Elim	Inversion	Focus

Constructor Patterns

$$A^+ \quad ::= \quad A^+ \oplus B^+ \mid A^+ \otimes B^+ \mid \downarrow A^- \mid X^+$$

$$\frac{\Delta \Vdash c :: A^+}{\Delta \Vdash \mathsf{inl}\, c :: A^+ \oplus B^+} \qquad \frac{\Delta \Vdash c :: B^+}{\Delta \Vdash \mathsf{inr}\, c :: A^+ \oplus B^+}$$

$$\frac{\Delta_1 \Vdash c_1 :: A^+ \quad \Delta_2 \Vdash c_2 :: B^+}{\Delta_1, \Delta_2 \Vdash (c_1, c_2) :: A^+ \otimes B^+}$$

$$\frac{}{x : A^- \Vdash x :: \downarrow A^-} \qquad \frac{}{x : X^+ \Vdash x :: X^+}$$

Destructor Patterns

$$\begin{array}{rcl} A^{\text{-}} & ::= & \uparrow A^{+} \mid A^{+} \to B^{\text{-}} \mid A^{\text{-}} \& B^{\text{-}} \\ \gamma & ::= & A^{+} \mid X^{\text{-}} \end{array}$$

$$\frac{}{\cdot \Vdash \epsilon :: \uparrow A^{+} > A^{+}} \qquad \frac{}{\cdot \Vdash \epsilon :: X^{\text{-}} > X^{\text{-}}}$$

$$\frac{\Delta_1 \Vdash c :: A^{+} \quad \Delta_2 \Vdash d :: B^{\text{-}} > \gamma}{\Delta_1, \Delta_2 \Vdash c\,; d :: A^{+} \to B^{\text{-}} > \gamma}$$

$$\frac{\Delta \Vdash d :: A^{\text{-}} > \gamma}{\Delta \Vdash \mathsf{fst};\ d :: A^{\text{-}} \& B^{\text{-}} > \gamma} \qquad \frac{\Delta \Vdash d :: B^{\text{-}} > \gamma}{\Delta \Vdash \mathsf{snd};\ d :: A^{\text{-}} \& B^{\text{-}} > \gamma}$$

Right Focus, Left Inversion

$$\alpha ::= X^+ \mid C^- \qquad \gamma ::= X^- \mid C^+$$
$$\Delta ::= \cdot \mid \Delta, x : \alpha \qquad \Gamma ::= \cdot \mid \Gamma, \Delta$$

$$\boxed{\Gamma \vdash v^+ :: C^+}$$

$$\frac{\Delta \Vdash c :: C^+ \quad \Gamma \vdash \sigma : \Delta}{\Gamma \vdash c\,[\sigma] :: C^+}$$

$$\boxed{\Gamma \vdash k^+ : \gamma_0 > \gamma}$$

$$\frac{}{\Gamma \vdash \epsilon : X^- > X^-}$$

$$\frac{\forall (\Delta \Vdash c :: C^+) : \quad \Gamma, \Delta \vdash \phi^+(c) : \gamma}{\Gamma \vdash \mathsf{cont}^+(\phi^+) : C^+ > \gamma}$$

Right Inversion, Left Focus

$$\boxed{\Gamma \vdash v^{\text{-}} : \alpha}$$

$$\frac{\forall(\Delta \Vdash d :: C^{\text{-}} > \gamma) : \quad \Gamma, \Delta \vdash \phi^{\text{-}}(d) : \gamma}{\Gamma \vdash \mathsf{val}^{\text{-}}(\phi^{\text{-}}) : C^{\text{-}}} \qquad \frac{x : X^{+} \in \Gamma}{\Gamma \vdash x : X^{+}}$$

$$\boxed{\Gamma \vdash k^{\text{-}} :: C^{\text{-}} > \gamma}$$

$$\frac{\Delta \Vdash d :: C^{\text{-}} > \gamma_0 \quad \Gamma \vdash \sigma : \Delta \quad \Gamma \vdash k^{+} : \gamma_0 > \gamma}{\Gamma \vdash d[\sigma]; k^{+} :: C^{\text{-}} > \gamma}$$

Neutral, Substitution

$$\boxed{\Gamma \vdash e : \gamma}$$

$$\frac{\Gamma \vdash v^{+} :: C^{+}}{\Gamma \vdash v^{+} : C^{+}} \qquad \frac{x : C^{-} \in \Gamma \quad \Gamma \vdash k^{-} :: C^{-} > \gamma}{\Gamma \vdash x \bullet k^{-} : \gamma}$$

$$\boxed{\Gamma \vdash \sigma : \Delta}$$

$$\frac{}{\Gamma \vdash \cdot : \cdot} \qquad \frac{\Gamma \vdash \sigma : \Delta \quad \Gamma \vdash v^{-} : C^{-}}{\Gamma \vdash \sigma, v^{-}/x : \Delta, x : C^{-}}$$

Cut

$$\frac{\Gamma \vdash v^{-} : C^{-} \quad \Gamma \vdash k^{-} :: C^{-} > \gamma}{\Gamma \vdash v^{-} \bullet k^{-} : \gamma} \qquad \frac{\Gamma \vdash v^{+} :: C^{+} \quad \Gamma \vdash k^{+} : C^{+} > \gamma}{\Gamma \vdash v^{+} \bullet k^{+} : \gamma}$$

$$\frac{\Gamma \vdash e : \gamma_0 \quad \Gamma \vdash k^{+} : \gamma_0 > \gamma}{\Gamma \vdash e \,;\, k^{+} : \gamma} \qquad \frac{\Gamma \vdash k^{-} :: C^{-} > \gamma_0 \quad \Gamma \vdash k^{+} : \gamma_0 > \gamma}{\Gamma \vdash k^{-} \,;\, k^{+} :: C^{-} > \gamma}$$

$$\frac{\Gamma \vdash k_1^{+} : \gamma_0 > \gamma_1 \quad \Gamma \vdash k_2^{+} : \gamma_1 > \gamma}{\Gamma \vdash k_1^{+} \,;\, k_2^{+} : \gamma_0 > \gamma}$$

Identity

$$\frac{}{\Gamma \vdash \epsilon : C^{+} > C^{+}}$$

$$\frac{\Delta \subseteq \Gamma}{\Gamma \vdash \mathsf{id} : \Delta}$$

$$\frac{x : C^{-} \in \Gamma}{\Gamma \vdash x : C^{-}}$$

Inconsistency

$$\frac{\Gamma, x : C^{\text{-}} \vdash v^{\text{-}} : C^{\text{-}}}{\Gamma \vdash \mathsf{fix}(x.v^{\text{-}}) : C^{\text{-}}}$$

Operational Semantics (Positive Cut)

$$\frac{\phi^+(c)\ \mathsf{defined}}{c\,[\sigma] \bullet \mathsf{cont}^+(\phi^+) \hookrightarrow \phi^+(c)\,[\sigma]}$$

$$\frac{}{v^+ \bullet (k_1^+\,;\, k_2^+) \hookrightarrow (v^+ \bullet k_1^+)\,;\, k_2^+}$$

$$\frac{}{v^+ \bullet \epsilon \hookrightarrow v^+}$$

Operational Semantics (Negative Cut)

$$\frac{\phi^{\text{-}}(d)\ \text{defined}}{\mathsf{val}^{\text{-}}(\phi^{\text{-}}) \bullet (d[\sigma]; k^{+}) \hookrightarrow (\phi^{\text{-}}(d)\,[\sigma])\,;\ k^{+}}$$

$$\frac{}{v^{\text{-}} \bullet (k^{\text{-}}\,;\ k^{+}) \hookrightarrow (v^{\text{-}} \bullet k^{\text{-}})\,;\ k^{+}}$$

$$\frac{}{\mathsf{fix}(x.v^{\text{-}}) \bullet k^{\text{-}} \hookrightarrow v^{\text{-}}\,[\mathsf{fix}(x.v^{\text{-}})/x] \bullet k^{\text{-}}}$$

Operational Semantics (Case)

$$\frac{e \hookrightarrow e'}{e\,;\,k^+ \hookrightarrow e'\,;\,k^+} \qquad \frac{}{v^+\,;\,k^+ \hookrightarrow v^+ \bullet k^+}$$

Patterns for Datatypes with Binding

Contextual Formula

Pos. formula	A^+	::=	$X^+ \mid \downarrow A^-$
			$\mid 1 \mid A^+ \otimes B^+ \mid 0 \mid A^+ \oplus B^+$
			$\mid P \mid R \Rightarrow A^+ \mid \Box A^+$
Rule	R	::=	$P \Leftarrow A_1^+ \Leftarrow \ldots \Leftarrow A_n^+$
Neg. formula	A^-	::=	$X^- \mid \uparrow A^+ \mid A^+ \rightarrow B^-$
			$\mid \top \mid A^- \& B^- \mid \nu\, X^-.A^-$
			$\mid R \curlywedge B^- \mid \diamond A^-$
Rule Context	Ψ	::=	$\cdot \mid \Psi, u : R$
CPF	C^+	::=	$\langle \Psi \rangle\, A^+$
CNF	C^-	::=	$\langle \Psi \rangle\, A^-$

Constructor Patterns

$$\frac{}{x : X^+ \,;\, \Psi \Vdash x :: X^+} \qquad \frac{}{x : \langle \Psi \rangle\, A^- \,;\, \Psi \Vdash x :: \downarrow A^-}$$

$$\frac{}{\cdot \,;\, \Psi \Vdash () :: 1} \qquad \frac{\Delta_1 \,;\, \Psi \Vdash p_1 :: A^+ \quad \Delta_2 \,;\, \Psi \Vdash p_2 :: B^+}{\Delta_1, \Delta_2 \,;\, \Psi \Vdash (p_1, p_2) :: A^+ \otimes B^+}$$

(no rule for 0)

$$\frac{\Delta \,;\, \Psi \Vdash p :: A^+}{\Delta \,;\, \Psi \Vdash \mathsf{inl}\, p :: A^+ \oplus B^+} \qquad \frac{\Delta \,;\, \Psi \Vdash p :: B^+}{\Delta \,;\, \Psi \Vdash \mathsf{inr}\, p :: A^+ \oplus B^+}$$

Constructor Patterns (Definitional Types)

$$\frac{u : P \Leftarrow A_1^+ \Leftarrow \ldots \Leftarrow A_n^+ \in (\Sigma, \Psi) \qquad \Delta_1 \,;\, \Psi \Vdash p_1 :: A_1^+ \quad \ldots \quad \Delta_n \,;\, \Psi \Vdash p_n :: A_n^+}{\Delta_1, \ldots, \Delta_n \,;\, \Psi \Vdash u\, p_1 \ldots p_n :: P}$$

$$\frac{\Delta \,;\, \Psi, u : R \Vdash p :: B^+}{\Delta \,;\, \Psi \Vdash \lambda\, u.\, p :: R \Rightarrow B^+} \qquad \frac{\Delta \,;\, \cdot \Vdash p :: A^+}{\Delta \,;\, \Psi \Vdash \mathsf{box}\, p :: \Box A^+}$$

Destructor Patterns

$$\frac{}{\cdot\,;\ \Psi \Vdash \epsilon :: X^{-} > X^{-}} \qquad \frac{}{\cdot\,;\ \Psi \Vdash \epsilon :: \uparrow A^{+} > \langle \Psi \rangle\ A^{+}}$$

$$\frac{\Delta_1\,;\ \Psi \Vdash p :: A^{+} \quad \Delta_2\,;\ \Psi \Vdash n :: B^{-} > \gamma}{\Delta_1, \Delta_2\,;\ \Psi \Vdash p\,; n :: A^{+} \to B^{-} > \gamma}$$

$$\frac{\Delta\,;\ \Psi \Vdash n :: A^{-} > \gamma}{\Delta\,;\ \Psi \Vdash \mathsf{fst};\ n :: A^{-} \& B^{-} > \gamma} \qquad \frac{\Delta\,;\ \Psi \Vdash n :: B^{-} > \gamma}{\Delta\,;\ \Psi \Vdash \mathsf{snd}; n :: A^{-} \& B^{-} > \gamma}$$

(no rule for $\top$)

$$\frac{\Delta\,;\ \Psi \Vdash n :: [\nu\, X^{-}.A^{-}/X^{-}]A^{-} > \gamma}{\Delta\,;\ \Psi \Vdash \mathsf{out};\ n :: \nu\, X^{-}.A^{-} > \gamma}$$

Destructor Patterns (Definitional)

$$\frac{\Delta\,;\,\Psi, u : R \Vdash n :: B^{-} > \gamma}{\Delta\,;\,\Psi \Vdash \mathsf{unpack};\; u.n :: R \curlywedge B^{-} > \gamma}$$

$$\frac{\Delta\,;\,\cdot \Vdash n :: A^{-} > \gamma}{\Delta\,;\,\Psi \Vdash \mathsf{undia};\; n :: \diamond A^{-} > \gamma}$$

Contextual Patterns

$$c \quad ::= \quad \overline{\Psi}.p$$
$$d \quad ::= \quad \overline{\Psi}.n$$

$$\boxed{\Delta \Vdash c :: \langle \Psi \rangle\, A^{+} \text{ and } \Delta \Vdash d :: \langle \Psi \rangle\, A^{+} > \gamma}$$

$$\frac{\Delta\,;\, \Psi \Vdash p :: A^{+}}{\Delta \Vdash \overline{\Psi}.p :: \langle \Psi \rangle\, A^{+}} \qquad \frac{\Delta\,;\, \Psi \Vdash n :: A^{-} > \gamma}{\Delta \Vdash \overline{\Psi}.n :: \langle \Psi \rangle\, A^{-} > \gamma}$$

Logical Properties

Shocking Equalities

Proposition 1 (“Shocking” equalities)**.**

1. $R \Rightarrow (A^+ \oplus B^+) \approx (R \Rightarrow A^+) \oplus (R \Rightarrow B^+)$

   (cf. $\forall x.(A \oplus B) \approx (\forall x.A) \oplus (\forall x.B)$*)*

2. $(R \curlywedge A^-)\&(R \curlywedge B^-) \approx R \curlywedge (A^-\&B^-)$

   (cf. $(\exists x.A)\&(\exists x.B) \approx \exists x.(A\&B)$*)*

Proposition 2 (Some/any)**.**

1. $\downarrow(R \curlywedge A^-) \approx R \Rightarrow \downarrow A^-$
2. $\uparrow(R \Rightarrow A^+) \approx R \curlywedge \uparrow A^+$

Examples

Example

Define

```
and* (true  , true ) = true[·]
and* (true  , false) = false[·]
and* (false , true ) = false[·]
and* (false , false) = false[·]
```

Then $\cdot \vdash \mathsf{cont}^{+}(\mathsf{and}*) : (\mathsf{bool} \otimes \mathsf{bool}) > \mathsf{bool}$

Example

$$e \quad ::= \quad \mathsf{num}[k] \mid e_1 \odot_f e_2 \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$$

Represent with a datatype ari:

$$\mathsf{zero} : \mathsf{nat},\ \mathsf{succ} : \mathsf{nat} \Leftarrow \mathsf{nat},$$
$$\mathsf{num} : \mathsf{ari} \Leftarrow \mathsf{nat}$$
$$\mathsf{binop} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{nat} \otimes \mathsf{nat} \to \mathsf{nat}) \Leftarrow \mathsf{ari}$$
$$\mathsf{let} : \mathsf{ari} \Leftarrow \mathsf{ari} \Leftarrow (\mathsf{ari} \Rightarrow \mathsf{ari})$$

Example

Evaluator:

$$\cdot \vdash \mathsf{fix}(ev.\mathit{ev}^*) : \langle \Psi_{\mathsf{ari}} \rangle \, (\mathsf{ari} \to \mathsf{nat})$$

STS:

$$\forall(\Delta \Vdash p :: \langle \Psi_{\mathsf{ari}} \rangle \, \mathsf{ari}).$$
$$(ev : \langle \Psi_{\mathsf{ari}} \rangle \, \mathsf{ari} \to \mathsf{nat}, \Delta) \vdash (\mathit{ev}^* \; p) : \langle \Psi_{\mathsf{ari}} \rangle \, \mathsf{nat}$$

Example

$$\forall(\Delta \Vdash p :: \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{ari}).$$
$$(ev : \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{ari} \to \mathsf{nat}, \Delta) \vdash (\mathit{ev}^*\ p) : \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{nat}$$

$$\begin{aligned}
\mathit{ev}^*\ &(\mathsf{num}\ p) &&\mapsto p \\
\mathit{ev}^*\ &(\mathsf{binop}\ p_1\ \ f\ \ p_2) &&\mapsto f\ (ev\ p_1)\ (ev\ p_2) \\
\mathit{ev}^*\ &(\mathsf{let}\ p_0\ \ (\lambda\, u.\, p)) &&\mapsto ev\ (\mathit{apply}\ (\lambda\, u.\, p, p_0))
\end{aligned}$$

$$\mathit{apply} : \langle \Psi_{\mathsf{ari}} \rangle\, (\mathsf{ari} \Rightarrow \mathsf{ari}) \to (\mathsf{ari} \to \uparrow \mathsf{ari})$$

Example

$$\forall(\Delta \Vdash p :: \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{ari}).$$
$$(ev : \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{ari} \to \mathsf{nat}, \Delta) \vdash (\mathit{ev}^*\ p) : \langle \Psi_{\mathsf{ari}} \rangle\, \mathsf{nat}$$

$$\begin{aligned}
\mathit{ev}^*\ &(\mathsf{num}\ p) &&\mapsto p \\
\mathit{ev}^*\ &(\mathsf{binop}\ p_1\ \ f\ \ p_2) &&\mapsto f\ (ev\ p_1)\ (ev\ p_2) \\
\mathit{ev}^*\ &(\mathsf{let}\ p_0\ \ (\lambda\, u.\, p)) &&\mapsto ev\ (\mathit{apply}\ (\lambda\, u.\, p, p_0))
\end{aligned}$$

$$\mathit{apply} : \langle \Psi_{\mathsf{ari}} \rangle\, (\mathsf{ari} \Rightarrow \mathsf{ari}) \to (\mathsf{ari} \to \uparrow \mathsf{ari})$$