# Subtyping and Intersection Types Revisited

Frank Pfenning

Carnegie Mellon University

International Conference on Functional Programming (ICFP'07)

Freiburg, Germany, October 1-3, 2007

Joint work with Rowan Davies, Joshua Dunfield, William Lovas, and Noam Zeilberger

*Work in progress!*

# Outline

- Verificationist and pragmatist meaning theory
- Canonical proofs and atomic subtyping
- Identity and substitution
- Defining higher-order subtyping
- Sound and complete rules for higher-order subtyping
- Intersections
- Higher-order subtyping extended
- Monads: the bridge to functional programming
- The value restriction and distributivity

# What is Logic About?

- Not truth, but *consequence*
- Not particular propositions, but *arbitrary* propositions

| socrates is a man<br>all men are mortal | $P(c)\ \mathit{true}$<br>$\forall x.\, P(x) \supset Q(x)\ \mathit{true}$ |
|---|---|
| socrates is mortal | $Q(c)\ \mathit{true}$ |

$$\frac{P(c)\ \mathit{true} \qquad \forall x.\, P(x) \supset Q(x)\ \mathit{true}}{Q(c)\ \mathit{true}}$$

- Proofs are *parametric* in atomic propositions
- If $A\ \mathit{true}$ then $[B/P]A\ \mathit{true}$ for any $B$

# The Meaning of Propositions

- Atomic propositions are parameters
- Meaning of compound propositions should be determined by their constituents
- *Verificationist*: meaning of a proposition is determined by its verifications

$$\frac{A\ true \quad B\ true}{A \,\&\, B\ true}\ \&I$$

- Introduction rules define: read bottom-up
- Elimination rules are justified from introduction rules

# The Meaning of Propositions

- *Pragmatist*: meaning of a proposition is determined by its uses

$$\frac{A \mathbin{\&} B \ true}{A \ true} \&E_1 \qquad \frac{A \mathbin{\&} B \ true}{B \ true} \&E_2$$

- Elimination rules define: read top-down
- Introduction rules are justified from elimination rules

# An Aside

- Verificationist = ML programmer (= call-by-value)
- Pragmatist = Haskell programmer (= call-by-name)
- See [Zeilberger POPL'08]
- Origins in [Dummett'76] [Martin-Löf'83]

# Hypothetical Reasoning

- Hypothetical reasoning from a notion of consequence

$$\dfrac{\dfrac{A \mathbin{\&} B \; \mathit{true}}{B \; \mathit{true}}\,\&E_1 \qquad \dfrac{A \mathbin{\&} B \; \mathit{true}}{A \; \mathit{true}}\,\&E_2}{B \mathbin{\&} A \; \mathit{true}}\,\&I$$

- Written as hypothetical judgment

$$A \mathbin{\&} B \; \mathit{true} \vdash B \mathbin{\&} A \; \mathit{true}$$

# Canonical Proofs

- Verificationist to establish conclusion
- Pragmatist to use hypotheses
- Meet at an atomic proposition $P$

$$\frac{A\ \mathit{verif} \quad B\ \mathit{verif}}{A \,\&\, B\ \mathit{verif}}\,\&I \qquad \frac{P\ \mathit{use} \quad P = Q}{Q\ \mathit{verif}}$$

$$\frac{A \,\&\, B\ \mathit{use}}{A\ \mathit{use}}\,\&E_1 \qquad \frac{A \,\&\, B\ \mathit{use}}{B\ \mathit{use}}\,\&E_2$$

- Bidirectional subformula property

# Implication

- Implication requires hypothetical reasoning

$$\dfrac{\dfrac{}{A\ \mathit{use}}\,x \quad \vdots \quad B\ \mathit{verif}}{A \supset B\ \mathit{verif}} \supset I^x \qquad \dfrac{A \supset B\ \mathit{use} \quad A\ \mathit{verif}}{B\ \mathit{use}}$$

- Maintains bidirectional subformula property

# Harmony

- *Identity*

  $A\ \mathit{use} \vdash A\ \mathit{verif}$ for any proposition $A$

  - Assume for atomic propositions (= parameters)
  - Ensure for compound propositions

- *(Hereditary) Proof Substitution*

  If $\Gamma, A\ \mathit{use} \vdash C\ \mathit{verif}$
  and $\Gamma \vdash A\ \mathit{verif}$
  then $\Gamma \vdash C\ \mathit{verif}$

  - Hereditary substitution to obtain canonical proofs

# Relating Atomic Propositions

- Relate (otherwise parametric) atomic propositions
- Based on entailment

$$\begin{array}{ccc} \text{man} & \leq & \text{mortal} \\ P & \leq & Q \qquad (P \vdash Q) \end{array}$$

- $P \leq Q$ is an *assumption* on parameters $P$ and $Q$
- Use when logical connectives have been eliminated

$$\frac{P\ use \quad P = Q}{Q\ verif} \longrightarrow \frac{P\ use \quad P \leq Q}{Q\ verif}$$

# Rules of Atomic Subtyping

- Reflexivity, by identity $P\ use \vdash P\ verif$

$$\frac{}{P \leq P}$$

- Transitiviy, by substitution from $P\ use \vdash Q\ verif$ and $Q\ use \vdash R\ verif$

$$\frac{P \leq Q \quad Q \leq R}{P \leq R}$$

# Curry-Howard-DeBruijn

- Propositions as types; proofs as terms
- Annotate judgments with proof terms
  - $A\ \mathit{verif}$ becomes $M \Leftarrow A$ (check $M$ against $A$)
  - $A\ \mathit{use}$ becomes $R \Rightarrow A$ ($R$ synthesizes $A$)
- Canonical proofs as canonical terms ($\beta$-normal, $\eta$-long)

$$\frac{\Gamma, x \Rightarrow A \vdash M \Leftarrow B}{\Gamma \vdash \lambda x.\, M \Leftarrow A \to B} \to I \qquad \frac{\Gamma \vdash R \Rightarrow P \quad P \leq Q}{\Gamma \vdash R \Leftarrow Q}$$

$$\frac{}{\Gamma, x \Rightarrow A \vdash x \Rightarrow A} \qquad \frac{\Gamma \vdash R \Rightarrow A \to B \quad \Gamma \vdash M \Leftarrow A}{\Gamma \vdash R\, M \Rightarrow B} \to E$$

# Identity Revisited

- Canonical term grammar

$$\begin{array}{llll} \text{Canonical terms} & M & ::= & \lambda x.\, M \mid R \\ \text{Atomic terms} & R & ::= & x \mid R\, M \end{array}$$

- Identity with terms

  $x \Rightarrow A \vdash \eta_A(x) \Leftarrow A$ for any proposition $A$

- $\eta_A(R)$ defined by induction on $A$

$$\begin{array}{lll} \eta_P(R) & = & R \\ \eta_{A \to B}(R) & = & \lambda x.\, \eta_B(R\, \eta_A(x)) \end{array}$$

- Definition extends modularly to new connectives

# Substitution Revisited

- Hereditary substitution with terms

  If $\Gamma \vdash M \Leftarrow A$ and $\Gamma, x \Rightarrow A \vdash N \Leftarrow C$

  then $\Gamma \vdash [M/x]_A N \Leftarrow C$

- Define $[M/x]_A N$ by nested induction on $A$ and $N$

$$
\begin{array}{lcll}
[M/x]_A(\lambda y.\, N) & = & \lambda y.\, [M/x]_A N & \\
[M/x]_A(R) & = & [\![M/x]\!]_A R & \text{if } \mathrm{hd}(R) = x \\
[M/x]_A(R) & = & [M/x]^r_A R & \text{if } \mathrm{hd}(R) \neq x \\
[M/x]^r_A(R\, N) & = & ([M/x]^r_A R)\, ([M/x]_A N) & \\
[M/x]^r_A(y) & = & y &
\end{array}
$$

# Hereditary Substitution

- Call $[\![M/x]\!]_A(R)$ when $\mathrm{hd}(R) = x$
- Returns canonical term and its type $M' : A'$ with $A'$ a constituent of $A$

$$
\begin{array}{lcl}
[\![M/x]\!]_A(x) & = & M : A \\
[\![M/x]\!]_A(R\,N) & = & [N'/y]_B M' : C \\
& & \text{where } [M/x]_A N = N' \\
& & \text{and } [\![M/x]\!]_A R = \lambda y.\, M' : B \to C
\end{array}
$$

- Refers back to ordinary substitution with smaller type

# Application: Logical Frameworks

- LF Logical Framework
- Based on dependent types $\lambda^{\Pi}$
  - Higher-order abstract syntax
  - Judgments as types
- Object language and rules are encoded as signatures with constant declarations
- Theory of subtyping presented here extends to dependent types [Lovas & Pf'07]
  - Allow declarations $a \leq b$ for type families $a$ and $b$.
  - A little more later in this talk

# Example: Encoding CBV

- Signature (with $\mathsf{exp} : \mathsf{type}$ and $\mathsf{val} : \mathsf{type}$)

$$
\begin{array}{lcl}
\mathsf{lam} & : & (\mathsf{val} \longrightarrow \mathsf{exp}) \longrightarrow \mathsf{val} \\
\mathsf{app} & : & \mathsf{exp} \longrightarrow \mathsf{exp} \longrightarrow \mathsf{exp} \\
\mathsf{val} & \leq & \mathsf{exp}
\end{array}
$$

- Sample derivation

$$
\dfrac{\mathsf{lam} \Rightarrow (\mathsf{val} \longrightarrow \mathsf{exp}) \longrightarrow \mathsf{val} \qquad \dfrac{\dfrac{\dfrac{}{x \Rightarrow \mathsf{val} \vdash x \Rightarrow \mathsf{val}} \quad \mathsf{val} \leq \mathsf{exp}}{x \Rightarrow \mathsf{val} \vdash x \Leftarrow \mathsf{exp}}}{\lambda x.\, x \Leftarrow \mathsf{val} \longrightarrow \mathsf{exp}}}{\mathsf{lam}\,(\lambda x.\, x) \Leftarrow \mathsf{val}}
$$

# Example: Encoding Polytypes

- Predicative polymorphism

$$\begin{array}{lcl} \mathsf{arrow} & : & \mathsf{simp\_tp} \rightarrow \mathsf{simp\_tp} \rightarrow \mathsf{simp\_tp} \\ \mathsf{forall} & : & (\mathsf{simp\_tp} \rightarrow \mathsf{poly\_tp}) \rightarrow \mathsf{poly\_tp} \\ \mathsf{simp\_tp} & \leq & \mathsf{poly\_tp} \end{array}$$

- Impredicative polymorphism

$$\begin{array}{lcl} \mathsf{arrow} & : & \mathsf{simp\_tp} \rightarrow \mathsf{simp\_tp} \rightarrow \mathsf{simp\_tp} \\ \mathsf{forall} & : & (\mathsf{poly\_tp} \rightarrow \mathsf{poly\_tp}) \rightarrow \mathsf{poly\_tp} \\ \mathsf{simp\_tp} & \leq & \mathsf{poly\_tp} \end{array}$$

# Summary So Far

- Meaning of connectives given by canonical proofs
- Analyze the structure of terms in two directions
- Bi-directional type-checking under CHdB isomorphism
- Identity and substitution principles
- Subtyping as entailment on atomic propositions
- Applicable to dependent types

# Subtyping at Higher Types

- What about the usual co-/contra-variant rule of subtyping?

$$\frac{B_1 \leq A_1 \quad A_2 \leq B_2}{A_1 \to A_2 \leq B_1 \to B_2}$$

- No need for such a rule
  - The logical meaning of $\to$ is already given
  - The logical meaning of $\leq$ is already given
- Canonical forms (specifically: $\eta$-long forms) are crucial

# Defining Higher-Order Subtyping

- We can *define* a notion of subtyping several ways

(1) $A \leq_1 B$ if for all $\Gamma \vdash M \Leftarrow A$ we have $\Gamma \vdash M \Leftarrow B$

(2) $A \leq_2 B$ if for all $\Gamma, x \Rightarrow B \vdash N \Leftarrow C$
we have $\Gamma, x \Rightarrow A \vdash N \Leftarrow C$

(3) $A \leq_3 B$ if $x \Rightarrow A \vdash \eta_A(x) \Leftarrow B$

(4) $A \leq_4 B$ if for all $\Gamma, x \Rightarrow B \vdash N \Leftarrow C$ and $\Gamma \vdash M \Leftarrow A$
we have $\Gamma \vdash [M/x]_B N \Leftarrow C$

- These are all equivalent!
- Only compare types of the same shape, since type shape determines term shape (cf. “refinement restriction”)

# Rules for Higher-Order Subtyping

- Rules so far for atomic types

$$\frac{}{P \leq P} \qquad \frac{P \leq Q \quad Q \leq R}{P \leq R}$$

- Also equivalent to (1)–(4) is adding the following rule

$$\frac{B_1 \leq A_1 \quad A_2 \leq B_2}{A_1 \rightarrow A_2 \leq B_1 \rightarrow B_2}$$

- General reflexivity and transitivity principles hold
- Mirror identity and substitution for entailment

# Subtyping Completeness

- The rules are complete in a strong sense:

  $A \leq B$ iff for all $\Gamma \vdash M \Leftarrow A$ we have $\Gamma \vdash M \Leftarrow B$!

- Possible due to the *open-ended* interpretation of subtyping
  - Quantification over $\Gamma$
  - Subtypings are stable under all extensions
  - Consistent with open-ended nature of LF
- Contrast with functional programming
  - Interested in *closed* terms for evaluation
  - Datatypes are inductive or recursive, not open-ended
  - Ironically, function spaces are open-ended
  - Cannot expect completeness

# Subtyping Alone is Insufficient

- ... in practice
- Consider types even and odd
- What is the type of successor?
- Need *intersection types*

$$\begin{array}{lcl} \mathsf{z} & : & \mathsf{even} \\ \mathsf{s} & : & (\mathsf{even} \longrightarrow \mathsf{odd}) \wedge (\mathsf{odd} \longrightarrow \mathsf{even}) \end{array}$$

- Expresses *multiple* properties of *one* term in a single type

# Meaning Explanations

- Resume verificationist ($\wedge I$) and pragmatist ($\wedge E_i$) programs

$$\frac{\Gamma \vdash M \Leftarrow A \quad \Gamma \vdash M \Leftarrow B}{\Gamma \vdash M \Leftarrow A \wedge B} \wedge I$$

$$\frac{\Gamma \vdash R \Rightarrow A \wedge B}{\Gamma \vdash R \Rightarrow A} \wedge E_1 \qquad \frac{\Gamma \vdash R \Rightarrow A \wedge B}{\Gamma \vdash R \Rightarrow B} \wedge E_2$$

- Type-theoretic, but not purely “logical”
- Again, no new subtyping rules, nothing else needed!
- Only rule concerned with subtyping remains the same

$$\frac{\Gamma \vdash R \Rightarrow P \quad P \leq Q}{\Gamma \vdash R \Leftarrow Q}$$

# Example: 2 is even

- Recall

$$\begin{aligned} \mathsf{z} &: \mathsf{even} \\ \mathsf{s} &: (\mathsf{even} \longrightarrow \mathsf{odd}) \wedge (\mathsf{odd} \longrightarrow \mathsf{even}) \end{aligned}$$

- To check $\mathsf{s}\,(\mathsf{s}\,\mathsf{z}) \Leftarrow \mathsf{even}$ we use
  - $\mathsf{s} \Rightarrow \mathsf{odd} \longrightarrow \mathsf{even}$ for the outer occurrence
  - $\mathsf{s} \Rightarrow \mathsf{even} \longrightarrow \mathsf{odd}$ for the inner occurrence
  - $\mathsf{z} \Rightarrow \mathsf{even}$ for the occurrence of $\mathsf{z}$

# Example: Open-Endedness

- Add type empty with no constructor
- We do not know that $\text{empty} \leq \text{even}$:
  - $x \Rightarrow \text{empty} \vdash x \Leftarrow \text{empty}$ but $x \Rightarrow \text{empty} \not\vdash x \Leftarrow \text{even}$ unless we *specify* $\text{empty} \leq \text{even}$.
- We do not know that $\text{even} \wedge \text{odd} \leq \text{empty}$ for a similar reason
- Currently there is no way to specify this [future work]

# Characterizing HO Subtyping

- The characterizations of subtyping from before are still equivalent. For example

(1) $A \leq_1 B$ if for all $\Gamma \vdash M \Leftarrow A$ we have $\Gamma \vdash M \Leftarrow B$

(3) $A \leq_3 B$ if $x \Rightarrow A \vdash \eta_A(x) \Leftarrow B$

- Several sound and complete set of rules for subtyping
  - Axiomatic
  - Sequent calculus

# Axiomatic Formulation I

- Reflexivity and transitivity

$$\frac{}{A \leq A} \qquad \frac{A \leq B \quad B \leq C}{A \leq C}$$

- Functions

$$\frac{B_1 \leq A_1 \quad A_2 \leq B_2}{A_1 \to A_2 \leq B_1 \to B_2}$$

# Axiomatic Formulation II

- Intersections

$$\frac{A \leq B \quad A \leq C}{A \leq B \wedge C} \qquad \frac{A \leq C}{A \wedge B \leq C} \quad \frac{B \leq C}{A \wedge B \leq C}$$

- Distributivity

$$\overline{(A \to B) \wedge (A \to C) \leq A \to (B \wedge C)}$$

# Sequent Calculus for Subtyping

- Use sequent calculus to show decidability
- Not necessary in an implementation!
  - Only to understand subtyping
  - Only atomic subtyping is required for type-checking
- Judgment $\Delta \leq C$

$$\frac{\Delta \leq A \quad \Delta \leq B}{\Delta \leq A \wedge B} \wedge R \qquad \frac{\Delta, A, B \leq C}{\Delta, A \wedge B \leq C} \wedge L$$

$$\frac{P \leq Q}{\Delta, P \leq Q} \qquad \frac{[A \leq A_i]_i \quad [B_i]_i \leq B}{\Delta, [A_i \to B_i]_i \leq A \to B}$$

# Algorithmic Typing

- Typing rules are non-deterministic

$$\frac{\Gamma \vdash R \Rightarrow A \wedge B}{\Gamma \vdash R \Rightarrow A} \qquad \frac{\Gamma \vdash R \Rightarrow A \wedge B}{\Gamma \vdash R \Rightarrow B}$$

- A “more efficient” system
  - $\Gamma \vdash R \Rrightarrow \Delta$ ($R$ has all types in $\Delta$)
  - $\Gamma \vdash M \Lleftarrow A$ ($M$ checks against $A$)
- Rules should maximally break down intersection
- Elide this refinement

# Algorithmic Typing Rules

$$\frac{\Gamma, x \Rightarrow A \vdash M \Leftarrow B}{\Gamma \vdash \lambda x.\, M \Leftarrow A \to B} \qquad \frac{\Gamma \vdash R \Rrightarrow \Delta, P \quad P \leq Q}{\Gamma \vdash R \Leftarrow Q}$$

$$\frac{\Gamma \vdash M \Leftarrow A \quad \Gamma \vdash M \Leftarrow B}{\Gamma \vdash M \Leftarrow A \wedge B}$$

$$\frac{}{\Gamma, x \Rightarrow A \vdash x \Rrightarrow A} \qquad \frac{\Gamma \vdash R \Rrightarrow \Delta, [A_i \to B_i]_i \quad [\Gamma \vdash M \Leftarrow A_i]_i}{\Gamma \vdash R\, M \Rrightarrow [B_i]_i}$$

$$\frac{\Gamma \vdash R \Rrightarrow \Delta, A, B}{\Gamma \vdash R \Rrightarrow \Delta, A \wedge B}$$

# Summary

- Logical meaning explanations via canonical proofs
- Simple and uniform system of subtyping and intersections
- Subtyping defined on atomic types only
- Intersection defined for synthesis and checking only
- Clear derived notions of subtyping for higher-order types
- Sound and complete set of rules
- Compare and intersect only types with compatible shape
  - Since type shape determines shape of canonical terms
- Applies to dependent types with morally identical rules

# Other Encoding Examples

- Barendregt's $\lambda$-cube
  - Uniform presentation of typing at all levels
  - Different corners of cube with different intersections
- Syntactic inclusions and properties
  - Values, expressions, and evaluation in functional languages
  - Hereditary Harrop formulas and logic programming
- Dependent properties of proofs
  - (Weak) (head) normal forms
  - Cut-free sequent proofs
  - Uniform sequent and natural deduction proofs

# Functional Programming

- Can we use this approach to design type systems for functional languages?
- Logical Framework (LF)
  - Complete rules for subtyping and intersection
  - *Open* interpretation of atomic types
  - *Closed* interpretation of function spaces
- Functional programming
  - *Closed* interpretation of atomic types
  - *Open* interpretation of function spaces
  - Canonical forms no longer fully characterize meaning
  - Operational semantics, non-termination, effects

# A Bridge

- Criterion (3) offers a bridge between logical frameworks and functional programming

(3) $A \leq_3 B$ if $x \Rightarrow A \vdash \eta_A(x) \Leftarrow B$

- Does not quantify over arbitrary terms or contexts
- $\eta$-expansions are available in functional language
- $\eta$-expansions are always canonical identity maps
- $\eta$-expansions depend only on type constructs in $A$ and $B$
- Stability under language extensions

# A Problem

- Some rules are *unsound* because they rely on the closed interpretation of function spaces (canonical forms, pure)
  - Intersection introduction requires a value restriction
  - Drop distributivity of $\wedge$ over $\rightarrow$
  - For counterexamples see [Davies & Pf. ICFP'00]
- Idea: use *monads* to isolate effects!
- Understand the effect of effects on subtyping and intersections

# Monads in Judgmental Form

- Maintain verificationist and pragmatist approach
- Meaning explanation by introduction and elimination forms
- Need new logical judgment
  - $\Gamma \vdash A\ lax$ (logically: lax truth of $A$)
- Type-theoretic version (with proof terms)
  - $\Gamma \vdash E \leftarrow A$

    $E$ is a potentially effectful computation of type $A$
  - Do not consider specific effects

# Judgmental Rules

- Relating lax truth to truth
- A pure term $M$ is a computation of type $A$

$$\frac{\Gamma \vdash M \Leftarrow A}{\Gamma \vdash M \leftarrow A}$$

- Substitution: we can compose computations $F$ before $E$

If $\Gamma \vdash F \leftarrow A$

and $\Gamma, x \Rightarrow A \vdash E \leftarrow C$

then $\Gamma \vdash \langle F/x \rangle_A E \leftarrow C$

# The Monad Type Constructor

- Type $\{A\}$ for pure terms denoting computations of type $A$
- Verificationist definition

$$\frac{\Gamma \vdash E \leftarrow A}{\Gamma \vdash \{E\} \Leftarrow \{A\}}\ \{\,\}I$$

- Pragmatist definition

$$\frac{\Gamma \vdash R \Rightarrow \{A\} \quad \Gamma, x \Rightarrow A \vdash E \leftarrow C}{\Gamma \vdash \text{let } \{x\} = R \text{ in } E \leftarrow C}\ \{\,\}E$$

# Identity and Substitution Revisited

- *Leftist* hereditary substitution $\langle E/x\rangle_A F$
- Defined by nested induction on $A$, $E$, and $F$

$$\begin{aligned}\langle \text{let } \{y\} = R \text{ in } E/x\rangle_A F &= \text{let } \{y\} = R \text{ in} \langle E/x\rangle_A F \\ \langle M/x\rangle_A E &= [M/x]_A E\end{aligned}$$

- Identity principle via $\eta$-expansion

$$\eta_{\{A\}}(R) = \{\text{let } \{x\} = R \text{ in } \eta_A(x)\}$$

# Subtyping and Intersections

- Rules for subtyping and intersection are not affected!
  - They were concerned with truth
  - Lax truth is a derived notion
- Orthogonality of language constructs pays off!
- It is *not* the case that

$$\frac{\Gamma \vdash E \leftarrow A \quad \Gamma \vdash E \leftarrow B}{\Gamma \vdash E \leftarrow A \wedge B} \textit{ wrong!}$$

# Higher-Order Subtyping

- Derived notions of subtyping remain unchanged
- New rule, axiomatically

$$\frac{A \leq B}{\{A\} \leq \{B\}}$$

# No Distributivity

- No distributivity: $\{A\} \wedge \{B\} \not\leq \{A \wedge B\}$
- Quick check via $\eta$-expansion

$$\dfrac{\dfrac{\dfrac{}{x \Rightarrow \{P\} \wedge \{Q\} \vdash x \Rightarrow \{P\} \wedge \{Q\}}}{x \Rightarrow \{P\} \wedge \{Q\} \vdash x \Rightarrow \{P\}} \qquad \dfrac{\overset{\textit{fails}}{y \Rightarrow P \vdash y \Leftarrow P \wedge Q}}{y \Rightarrow P \vdash y \leftarrow P \wedge Q}}{\dfrac{x \Rightarrow \{P\} \wedge \{Q\} \vdash \text{let } \{y\} = x \text{ in } y \leftarrow P \wedge Q}{x \Rightarrow \{P\} \wedge \{Q\} \vdash \{\text{let } \{y\} = x \text{ in } y\} \Leftarrow \{P \wedge Q\}}}$$

# Subtyping in Sequent Form

- In sequent form: commit

$$\frac{A \leq B}{\Delta, \{A\} \leq \{B\}}$$

- Contrast with functions

$$\frac{[A \leq A_i]_i \quad [B_i]_i \leq B}{\Delta, [A_i \to B_i]_i \leq A \to B}$$

  - Take all $i$ such that $A \leq A_i$

# The Value Restriction

- Call-by-value with effects requires some modifications
  - Value restriction on intersection introduction

$$\frac{\Gamma \vdash v : \sigma \quad \Gamma \vdash v : \tau}{\Gamma \vdash v : \sigma \wedge \tau}$$

  - No distributivity
- Subtyping at higher order (terms are not $\eta$-long)
  - Easy: we did all the work already!
- Typing annotations (terms are not $\beta$-normal)
  - Not entirely easy [Dunfield & Pf'03] [Dunfield'07]

# Deriving Restricted Rules

- Derive rules from standard encoding into monadic form

| | | | |
|---|---|---|---|
| Types | $\tau$ | $::=$ | $b \mid \tau_1 \rightarrow \tau_2$ |
| Computations | $e$ | $::=$ | $e_1\, e_2 \mid v$ |
| Values | $v$ | $::=$ | $x \mid \lambda x.\, e$ |

- Type translation

$$|b| = b$$
$$|\tau_1 \rightarrow \tau_2| = |\tau_1| \rightarrow \{\, |\tau_2| \,\}$$
$$|\tau_1 \wedge \tau_2| = |\tau_1| \wedge |\tau_2|$$

- Form of term translation

$|e| = E$ computations as monadic expressions

$|v| = M$ values as (pure) terms

# Value Restriction

- Properties of term translation (elided)
  - If $e : \tau$ then $|e| \leftarrow |\tau|$
  - If $v : \tau$ then $|v| \Leftarrow |\tau|$
- Value restriction since $\wedge$ does not distribute over $\{\ \}$.

$$\dfrac{\dfrac{\Gamma \vdash \{E\} \Leftarrow \{A\} \quad \Gamma \vdash \{E\} \Leftarrow \{B\}}{\Gamma \vdash \{E\} \Leftarrow \{A\} \wedge \{B\}}}{\Gamma \vdash \{E\} \Leftarrow \{A \wedge B\}} \text{ wrong!}$$

- Also recall: no rule to split $E \leftarrow A \wedge B$

# Distributivity

- Distributivity cannot be derived because

$$(A \to \{B\}) \wedge (A \to \{C\}) \leq A \to (\{B\} \wedge \{C\})$$

but

$$(A \to \{B\}) \wedge (A \to \{C\}) \not\leq A \to \{B \wedge C\}$$

- So

$$(\tau \to \sigma) \wedge (\tau \to \rho) \not\leq \tau \to (\sigma \wedge \rho)$$

# Other Considerations

- System has empty conjunction $\top$ (elided for this talk)
- Interpretation into type theory with proof irrelevance [mostly done]
  - Roughly: $M \Leftarrow A$ becomes $\langle M, N \rangle \Leftarrow \Sigma x{:}\check{A}.\, \hat{A}(x)$ where
  - $\check{A}$ is a type representing the shape of $A$
  - $\hat{A}(x)$ is a type family on elements of type $\check{A}$
  - $N \Leftarrow \check{A}(M)$ is a proof that $M$ has property $\check{A}$
  - $\langle M_1, N_1 \rangle = \langle M_2, N_2 \rangle$ iff $M_1 = M_2$ (by definition)
- Complexity due to definitional equality
- Coercion interpretation? [future work]

# And More

- Union types are complex, but fit the story [Zeilberger’07] [Dunfield’07] [Dunfield & Pf’03]
  - Co-value restriction on union types
- Parametric polymorphism is harder [Davies’05] [Dunfield]
- Deeper analysis of call-by-name and call-by-value
  - Judgmental method and focusing [Zeilberger POPL’08]
  - Positive and negative intersections, unions [Zeilberger’07]
  - Addings products, sums, and negations

# Summary: Origins

- Atomic propositions as parameters
- Atomic subtyping as assumptions
- Verificationist and pragmatist definitions of
    - logical connectives
    - canonical proofs
    - intersections
    - monads

# Summary: Destinations

- Exceedingly simple rules with subtyping and intersections
- Derived notion of higher-order subtyping
  - from substitution or $\eta$-expansion
  - with sound and complete sets of structural rules
- Richly epxressive logical framework $\lambda^{\Pi\leq\wedge}$
- Explanation of value restriction on intersections in call-by-value via monadic embedding

# All (Non-Derived) Rules

$$\frac{\Gamma, x \Rightarrow A \vdash M \Leftarrow B}{\Gamma \vdash \lambda x.\, M \Leftarrow A \rightarrow B} \rightarrow I \qquad \frac{\Gamma \vdash R \Rightarrow P \quad P \leq Q}{\Gamma \vdash R \Leftarrow Q}$$

$$\frac{\Gamma \vdash M \Leftarrow A \quad \Gamma \vdash M \Leftarrow B}{\Gamma \vdash M \Leftarrow A \wedge B} \wedge I$$

$$\frac{}{\Gamma, x \Rightarrow A \vdash x \Rightarrow A} \qquad \frac{\Gamma \vdash R \Rightarrow A \rightarrow B \quad \Gamma \vdash M \Leftarrow A}{\Gamma \vdash R\, M \Rightarrow B} \rightarrow E$$

$$\frac{\Gamma \vdash R \Rightarrow A \wedge B}{\Gamma \vdash R \Rightarrow A} \wedge E_1 \qquad \frac{\Gamma \vdash R \Rightarrow A \wedge B}{\Gamma \vdash R \Rightarrow B} \wedge E_2$$

$$\frac{}{P \leq P} \qquad \frac{P \leq Q \quad Q \leq S}{P \leq S}$$