

Reasoning About Languages with Binding

Can we do it yet?

Randy Pollack

LFCS, University of Edinburgh

Version of February 9, 2006

Outline

Needed: A challenge problem set

Approaches to Formalizing Binding

- First Order Representations

- Higher Order Abstract Syntax (HOAS)

- Weak HOAS

- Logics with “freshness”

Some Naturally Occurring Formalisation Problems

- Eigenvariables: Freshness of globally bound variables

- Simultaneous Substitution for Structural Recursion

- Infinite Structures

Summary

Outline

Needed: A challenge problem set

Approaches to Formalizing Binding

First Order Representations

Higher Order Abstract Syntax (HOAS)

Weak HOAS

Logics with “freshness”

Some Naturally Occurring Formalisation Problems

Eigenvariables: Freshness of globally bound variables

Simultaneous Substitution for Structural Recursion

Infinite Structures

Summary

Many technical approaches proposed to reason formally about languages with binding

- ▶ Concrete approaches
 - ▶ Curry/Feys, de Bruijn, Stoughton, McKinna/Pollack, Gordon/Melham, Hendriks/van Oostrum, ...
- ▶ Higher Order Abstract Syntax (HOAS): $(exp \rightarrow exp) \rightarrow exp$
 - ▶ LF (Harper, Honsell, Plotkin)
 - ▶ Hybrid (Ambler, Crole, Momigliano)
 - ▶ Twelf (Pfenning, Shürmann)
 - ▶ Miller/McDowell/Tiu
- ▶ Weak HOAS: $(atom \rightarrow exp) \rightarrow exp$
 - ▶ Context Calculus (Honsell, Miculan)
 - ▶ Desperoux/Felty/Hirschowitz
- ▶ “Freshness” approaches
 - ▶ FM, nominal (Pitts, Gabbay, Urban, Cheney)
 - ▶ Schöpp/Stark

Is it feasible to reason formally about languages with binding?

Can any of these approaches express all the definitions and arguments we want to use?

- ▶ Concrete approaches surely can, but at expense of long-winded reasoning.
- ▶ Can more convenient approaches do the job?
- ▶ We also want statements and proofs to be *natural*.

How can we know if an approach is adequate before investing significant effort in using it?

Needed: A challenge problem set for reasoning about binding

- ▶ In order to test technical approaches to reasoning about binding, a challenge problem set has been developed: POPLmark.
- ▶ Is it challenging enough?

Please suggest problems that give difficulty in one of the established techniques.

POPLmark Solutions; Jan 2006

CMU solution (Ashley-Rollman, Crary, and Harper) HOAS in Twelf.

All parts. First solution finished.

Stefan Berghofer Pure de Bruijn in Isabelle. All parts.

Xavier Leroy Locally nameless (McKinna/Pollack style) in Coq. Part 1a only. Other parts clearly possible, but very messy.

Jérôme Vouillon Pure de Bruijn in Coq. Parts 1 and 2.

Jevgenijs Sallinens Pure de Bruijn in Coq (following Vouillon's solution.) Parts 1a, 2a.

Aaron Stump Names for bound variables, be Bruijn *levels* for parameters. (The only solution where alpha equivalence wasn't the same as syntactic identity.)

Hongwei Xi In ATS. Only part 2a. Is part 1 possible?

Christian Urban et al. Nominal logic in Isabelle/HOL. Only part 1a. This is a brand new implementation. The only solution that comes with an implemented tool.

Outline

Needed: A challenge problem set

Approaches to Formalizing Binding

First Order Representations

Higher Order Abstract Syntax (HOAS)

Weak HOAS

Logics with “freshness”

Some Naturally Occurring Formalisation Problems

Eigenvariables: Freshness of globally bound variables

Simultaneous Substitution for Structural Recursion

Infinite Structures

Summary

First order representations

Clumsy, but can handle any informal structure and argument.

- ▶ Naive first order representation (Curry and Feys)
 - ▶ quotient by α -equivalence : feasible in an extensional logic?
 - ▶ substitution not structural
- ▶ Stoughton: named variables, simultaneous substitution.
 - ▶ simultaneous substitution is structural
- ▶ De Bruijn representation: nameless variables.
 - ▶ hard to work with, but there are developed libraries of lemmas
- ▶ McKinna/Pollack: distinct classes of free and bound names.
 - ▶ long-winded, but concrete and structural
- ▶ Locally nameless, Gordon/Melham, Buchholz
 - ▶ The best concrete approach?
- ▶ Adbmal (Hendriks/van Oostrum)
 - ▶ Good for reasoning about efficient implementation?

Unitary Substitution Not Structural

- ▶ $[-/-]b$ is defined by recursion on *length* of b ,
 - ▶ not on *structure* of b , since $[z/y]b$ is not a subterm of $\lambda y.b$.

$$[c/x]y \quad := \quad \text{if } x = y \text{ then } c \text{ else } y$$

$$[c/x](b_1 b_2) := ([c/x]b_1) ([c/x]b_2)$$

$$[c/x](\lambda y.b) := \lambda z.[c/x][z/y]b \quad z \text{ sufficiently fresh}$$

Simultaneous Substitution is Structural

$$x\rho \quad := \quad \rho x$$

$$(a b)\rho \quad := \quad (a\rho) (b\rho)$$

$$(\lambda x.b)\rho := \lambda z.(b(\rho, x=z)) \quad z \text{ sufficiently fresh}$$

- ▶ The choice of fresh z can be canonical;
 - ▶ then applying any substitution alpha-normalizes (Stoughton).

A *Locally Nameless* Representation of Expressions

- ▶ Two classes of variables:
 - ▶ Parameters (“free” variables): an infinite type of names with decidable equality.
 - ▶ de Bruijn indexes (for bound variables) are natural numbers.

```

Inductive exp : Set :=
| vp : par -> exp          (* free variables *)
| vv : nat -> exp         (* locally bound *)
| vlam : exp -> exp
| vapp : exp -> exp -> exp.
  
```

- ▶ The intention is that only de Bruijn - closed expressions are syntactically correct.
- ▶ Operations of *replacement* are structural. (No de Bruijn lifting.)
 - ▶ $psub : par \rightarrow exp \rightarrow exp$; replace a parameter by a term.
 - ▶ $vsub : exp \rightarrow exp$; replace index 0 by a term.

Syntactically correct locally nameless terms

An inductively defined predicate: no free indexes.

$$\frac{}{vclosed\ p} \quad \frac{vclosed\ u \quad vclosed\ v}{vclosed\ (app\ u\ v)} \quad \frac{vclosed\ (vsub\ p\ u)}{vclosed\ (lam\ u)}$$

- ▶ Instead of inducting over term structure, we induct over a derivation that the term is *vclosed*.
- ▶ Thus no case for free indexes arises when reasoning by induction over *vclosed* terms.

Why This Representation?

- ▶ Terms are equal up to α -equivalence .
 - ▶ In extensional logics (e.g. HOL) the correctness predicate *vclosed* can be made into a type.
- ▶ Substitution is simple and structural.
 - ▶ Since correct terms are de Bruijn closed, substitution does *not* require lifting of free indexes.
 - ▶ Since parameters are not bound, substitution can never capture parameters.
- ▶ Concrete treatment of free variables.
 - ▶ Natural statement of definitions and lemmas.
 - ▶ Supports explicit contexts, simultaneous substitution, environments
- ▶ **But there is no uniform treatment of eigenvariables.**
- ▶ **Seems to need generalised inductive definition.**

Gordon/Melham/Buchholz representation

- ▶ Built on top of locally nameless representation.
- ▶ A derived operation of named abstraction.
 - ▶ $\lambda x.t$ defined to be “de Bruijn abstraction of t after replacing every occurrence of x with index 0”.
- ▶ Then define an extensional subtype of proper terms (*vclosed*).
 - ▶ Only possible in extensional logic.
- ▶ Implementation: Gordon/Melham axioms for alpha-conversion.
 - ▶ Extensive examples by Michael Norrish in HOL.
- ▶ Also see recent note by Wilfried Buchholz on this approach.
- ▶ **Still no uniform treatment of eigenvariables.**

Higher Order Abstract Syntax (HOAS)

- ▶ Meta-type of binder: $(exp \rightarrow exp) \rightarrow exp$.
- ▶ Expressions are not inductively defined: positivity requirements.
- ▶ Can any HOAS approach handle simultaneous substitution?

Some examples:

- ▶ Pure logical framework (Edinburgh LF).
 - ▶ No induction/recursion over object structure.
- ▶ Logical frameworks with modalities, schema checking,
 - ▶ Twelf (Pfenning/Schürmann)
 - ▶ Some induction/recursion over object structure.
- ▶ Less pure LF, such as Isabelle.
 - ▶ No induction/recursion over object structure.
- ▶ *Hybrid* (Ambler/Crole/Momigliano)
- ▶ Stratified (several level) LF, e.g. Miller/McDowell, Felty.

Weak HOAS

- ▶ Datatype of expressions $e ::= var \mid e @ e \mid lam(var \rightarrow e)$.
- ▶ This type is not faithful because of “exotic” terms:

$$lam(\lambda v. \text{if } v = v_0 \text{ then } 2 \text{ else } 3)$$

- ▶ Exotic terms must be removed by some logical means.
- ▶ Substitution is a user-defined relation.

Some examples:

- ▶ Honsell/Miculan context calculus.
 - ▶ axiomatic
- ▶ Desperoux/Felty/Hirschowitz.
 - ▶ validity predicate

These approaches get complicated in practice.

Also, complicated justification.

Logics with “freshness”

- ▶ Pitts/Gabbay/Urban/Cheney nominal based approaches.
 - ▶ care must be taken with AC
 - ▶ new working implementation (Urban)
- ▶ Fiore/Plotkin/Turi abstract syntax.
 - ▶ not implemented
- ▶ Work by Ian Stark and Ulrich Shöpp (CSL '04, Shöpp thesis).
 - ▶ Gabbay/Pitts idea, with dependent types
 - ▶ not yet developed, but very interesting!
- ▶ $FO\lambda^{\Delta\nabla}$ of Miller/Tiu (TOCL).
 - ▶ not seriously implemented

Outline

Needed: A challenge problem set

Approaches to Formalizing Binding

First Order Representations

Higher Order Abstract Syntax (HOAS)

Weak HOAS

Logics with “freshness”

Some Naturally Occurring Formalisation Problems

Eigenvariables: Freshness of globally bound variables

Simultaneous Substitution for Structural Recursion

Infinite Structures

Summary

Eigenvariables: Freshness of globally bound variables

Simply typed lambda terms

- ▶ Let A, B, \dots be **simple types** (implicational propositions).
- ▶ x, y, z, \dots are variables.
- ▶ Define terms (a, b, c, \dots):

$$a ::= x \mid \lambda x. b \mid (ab).$$

- ▶ *Valid contexts* (Γ, Δ) are lists of uniquely labelled assumptions:
 - ▶ $x_1:A_1, \dots, x_n:A_n$ where the x_i are pairwise disjoint.
- ▶ Define **subcontext**:

$$\Gamma \sqsubseteq \Delta \quad \text{iff} \quad \forall x, A. x:A \in \Gamma \implies x:A \in \Delta$$

Δ contains every assumption occurring in Γ .

Assignment of simple types

As usual ...

ass $\Gamma \vdash x : A$

Γ valid, $x:A \in \Gamma$

elim
$$\frac{\Gamma \vdash c : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash ca : B}$$

intro
$$\frac{\Gamma, x:A \vdash [x/y]b : B}{\Gamma \vdash \lambda y.b : A \rightarrow B}$$

$x \notin b$

Problem: Prove weakening for this judgement

Lemma (Weakening)

$$\Gamma \vdash a : A \wedge \Gamma \sqsubseteq \Delta \wedge \Delta \text{ valid} \implies \Delta \vdash a : A .$$

Remark (de Bruijn notation precludes natural statements)

- ▶ *If we use de Bruijn indexes for free (global) variables, neither the definition of \sqsubseteq nor the statement of the lemma take the natural forms given above.*
- ▶ *Permuting the context requires lifting free indexes.*
- ▶ *Even more messy with dependent types.*

Prove weakening

$$\Gamma \vdash a : A \wedge \Gamma \sqsubseteq \Delta \wedge \Delta \text{ valid} \implies \Delta \vdash a : A.$$

Proof: Attempt proof by induction on the derivation of $\Gamma \vdash a : A$

- ▶ Consider case for rule *intro*:
$$\frac{\Gamma, x:A \vdash [x/y]b : B \quad x \notin b}{\Gamma \vdash \lambda y.b : A \rightarrow B}$$

- ▶ By rule *intro* it suffices to show

$$\Delta, x:A \vdash [x/y]b : B \quad \text{for any } x \notin b.$$

- ▶ We have IH:

$$\forall \Phi. (\Gamma, x_0:A \sqsubseteq \Phi \wedge \Phi \text{ valid}) \implies \Phi \vdash [x_0/y]b : B$$

for some particular $x_0 \notin b$.

- ▶ It seems we want to instantiate Φ in IH with $\Delta, x_0:A \dots$
- ▶ ... but $\Delta, x_0:A$ may not be valid, as x_0 may occur in Δ .

Proof of weakening (contd)

This proof by Pitts; adopted by Norrish in a concrete setting.

- ▶ Let $(x\ y) \cdot b$ mean *permute all occurrences of x and y in b* .
- ▶ As a lemma (**equivariance**), show

$$\Gamma \vdash a : A \implies \forall x\ y. (x\ y) \cdot \Gamma \vdash (x\ y) \cdot a : A. \quad (1)$$

- ▶ Now, pick $z \notin x, \Delta, b$. It suffices to show

$$\Delta, z:A \vdash [z/y]b : B$$

which, by (1) is equivalent to

$$(z\ x_0) \cdot (\Delta, z:A) \vdash (z\ x_0) \cdot ([z/y]b) : B$$

which follows from IH.

We have a nice proof; what's the problem?

- ▶ We have to prove the equivariance property for every new judgement.
 - ▶ For some judgements, it isn't as easy as this example.
- ▶ This is a meta-fact; it doesn't really depend on the particular judgement.
- ▶ The meta-logic should handle equivariance uniformly ...
 - ▶ ... for example, Gabbay/Pitts/Urban, or Miller/Tiu.

But even if we use a meta-logic that proves equivariance uniformly ...

- ▶ ... we still use swapping explicitly (as in the *weakening* proof above) to handle each example where eigenvariable problems appear.

Better: we can package this swapping reasoning once and for all.

- ▶ This technique from McKinna/Pollack (1993).

A more uniform solution to eigenvariable problems

The following judgements are equivalent:

$$\text{ass} \quad \Gamma \vdash x : A \qquad \Gamma \text{ valid, } x:A \in \Gamma$$

$$\text{elim} \quad \frac{\Gamma \vdash c : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash ca : B}$$

$$\text{intro} \quad \frac{\Gamma, x:A \vdash [x/y]b : B}{\Gamma \vdash \lambda y.b : A \rightarrow B} \quad x \notin b$$

$$\text{ass} \quad \Gamma \Vdash x : A \qquad \Gamma \text{ valid, } x:A \in \Gamma$$

$$\text{elim} \quad \frac{\Gamma \Vdash c : A \rightarrow B \quad \Gamma \Vdash a : A}{\Gamma \Vdash ca : B}$$

$$\text{intro} \quad \frac{\forall x. x \notin \Gamma \implies \Gamma, x:A \Vdash [x/y]b : B}{\Gamma \Vdash \lambda y.b : A \rightarrow B}$$

Proof?

Lemma $\Gamma \Vdash a : A \implies \Gamma \vdash a : A$.

- ▶ Proof direct by induction on the derivation of $\Gamma \Vdash a : A$.

Lemma $\Gamma \vdash a : A \implies \Gamma \Vdash a : A$.

- ▶ Attempt proof by induction on the derivation of $\Gamma \vdash a : A$.
 - ▶ Consider the case of rule *intro*.
 - ▶ Any derivation of \vdash will use a particular variable, say x_0 .
 - ▶ The IH for this case is

$$\Gamma, x_0:A_0 \Vdash [x_0/y]b : B \quad (x_0 \notin b)$$

but to use the *intro* rule for \Vdash we need the premise

$$\forall x. x \notin \Gamma \implies \Gamma, x:A_0 \Vdash [x/y]b : B$$

- ▶ We cannot reason from a particular variable to all variables!

Proof continued: $\Gamma \vdash a : A \implies \Gamma \Vdash a : A$

We solve the problem using swapping, as in the *weakening* proof.

- ▶ As a lemma, show **equivariance** of \Vdash :

$$\Gamma \Vdash a : A \implies \forall x y. (x y) \cdot \Gamma \Vdash (x y) \cdot a : A. \quad (2)$$

- ▶ Now, pick $x \notin \Gamma$. From IH and (2) have

$$(x x_0) \cdot (\Gamma, x_0 : A_0) \Vdash (x x_0) \cdot ([x_0/y]b) : B$$

i.e. $\Gamma, x : A_0 \Vdash [x/y]b : B$ as required.

Why are we interested in this equivalence?

- ▶ *Weakening* for \Vdash is easily proved by induction on derivations ...
 - ▶ ... hence, equivalently, weakening for \vdash .
- ▶ Similarly, many eigenvariable problems are handled by induction on derivations of \Vdash .
- ▶ **The equivalence of \vdash and \Vdash “packages” the eigenvariable reasoning that we need for all examples I know of.**
 - ▶ Introduction of \vdash is easy: only need property for one fresh variable.
 - ▶ Elimination of \Vdash is powerful: get the IH for all sufficiently fresh variables.

But do we need two forms, with an ad hoc equivalence proof, for every judgement we define?

Definition using a *freshness* quantifier

Do we need two forms, with an ad hoc equivalence proof, for every judgement we define?

- ▶ In a nominal approach we hope to define one judgement using a freshness quantifier:

$$\frac{\text{fresh } x. \Gamma, x:A \vdash [x/y]b : B}{\Gamma \vdash \lambda y.b : A \rightarrow B}$$

- ▶ If this can be made to work, we get both the easy introduction rules of \vdash , and the strong IH of \Vdash , uniformly from the meta-logic.

Simultaneous Substitution

To allow definition by structural recursion,
rather than by well founded recursion.

A PER semantics for typing judgements of a Logical Framework

From Coquand/Pollack/Takeyama,
Fundamenta Informaticae 65(1), 2005

- ▶ Pure λ -terms as above $a, b, M, N \dots$
- ▶ A syntax of dependent types:

$$A, B ::= EI M \mid \text{fun } x^A . B \mid \star$$

- ▶ Objects in \star are “names” of types.
- ▶ For $M : \star$, $EI M$ is the type named by M .

Informal example: $\lambda a . \lambda x . x : \text{fun } a^{\star} . (\text{fun } x^{EI a} . EI a)$

even more informally: $\lambda a . \lambda x . x : \text{fun } a^{\star} . (EI a \rightarrow EI a)$

Outline: Semantics of Categorical Judgements (judgements from no assumptions)

- ▶ Simultaneously define:
 1. Type equality: $A = B$ is a PER on syntactic types,
 - ▶ Write $A \in \mathbf{Type}$ for $A = A$.
 2. Interpretation of a type: for $A \in \mathbf{Type}$, \bar{A} is a PER on objects.
 - ▶ Write $M = N : \bar{A}$.
 - ▶ Write $M : \bar{A}$ for $M = M : \bar{A}$.
- ▶ The definition is parameterised on base cases:
 - ▶ a given PER, $\bar{*}$, interpreting $*$,
 - ▶ a given family, \mathcal{E} over $\bar{*}$, ($\mathcal{E}(u)$ interprets $El\ u$).
 - ▶ These must have a property *saturated* (see the paper).

The point for this talk: **use of simultaneous substitutions to make the definition of \bar{A} structural.**

Informal definition of categorical semantics

Three cases in the definition:

1. $\bar{\star} = \star$ where $\bar{\star}$ is the given PER.

2. $\frac{M = N : \bar{\star}}{EI M = EI N}$ where $\overline{EI M}$ is $\mathcal{E}(M)$.

3. $\frac{A_1 = A_2 \quad M_1 = M_2 : \overline{A_1} \implies B_1[M_1] = B_2[M_2]}{\text{fun } x_1^{A_1}.B_1 = \text{fun } x_2^{A_2}.B_2}$

where $F_1 = F_2 : \overline{\text{fun } x^A.B}$ iff

$$M_1 = M_2 : \overline{A} \implies F_1 M_1 = F_2 M_2 : \overline{B[M_1]}.$$

How to formalize this definition?

Problems:

- ▶ induction-recursion not available in current proof tools
- ▶ occurrence of $B[N_1]$ is not a substructure of $\text{fun } x^A.B$ in:

$$F_1 = F_2 : \overline{\text{fun } x^A.B} \text{ iff } N_1 = N_2 : \bar{A} \implies F_1 N_1 = F_2 N_2 : \overline{B[N_1]}.$$

Solution: Unwind induction-recursion into

- ▶ first, a recursive definition of \bar{A} ...
- ▶ ... parameterized by a simultaneous substitution to make the definition structural;
 - ▶ We could use well-founded recursion to define \bar{A} ,
 - ▶ but that is *much* harder to reason about in intensional type theory.
- ▶ second, an inductive definition of $A = B$.

First, the interpretation of a syntactic type

Binary relation \bar{A} defined by recursion on syntactic structure of A .

- ▶ To make this definition structural, parameterise it by a substitution.

$$\begin{aligned} \overline{\star, \sigma} &:= \overline{\star} \\ \overline{EIM, \sigma} &:= \mathcal{E}(M\sigma) \\ \overline{\text{fun } x^A.B, \sigma} &:= F_1, F_2 \mapsto \\ &N_1 = N_2 : \overline{A\sigma} \implies F_1 N_1 = F_2 N_2 : \overline{B, (\sigma, x=N_1)} \\ \overline{A} &:= \overline{A, \sigma_{\text{id}}} \end{aligned}$$

- ▶ Formalisation must support simultaneous substitution ...
... can any variant of HOAS handle that?

\bar{A} is not yet a PER; later show that if $A \in \mathbf{Type}$ then \bar{A} is a PER.

Second, the relation of being equal (correct) types

Defined by induction.

$$\frac{}{\star = \star} \qquad \frac{M = N : \star}{EI\ M = EI\ N}$$

$$\frac{A_1 = A_2 \quad M_1 = M_2 : \overline{A_1} \implies B_1[M_1] = B_2[M_2]}{\text{fun } x_1^{A_1}.B_1 = \text{fun } x_2^{A_2}.B_2}$$

Finally, correctness

Lemma If $A = B$ then

- ▶ $\overline{A} = \overline{B}$ (extensionally)
- ▶ \overline{A} is a PER on objects
- ▶ $- = -$ is a PER on syntactic types

Infinite Structures

Outline: Hypothetical judgements (from assumptions)

- ▶ Introduce *environments*, ρ , *contexts*, C .
- ▶ Define a judgement $\rho_1 = \rho_2 : C$.
- ▶ Simultaneously define hypothetical judgements
 - ▶ C valid ,
 - ▶ $A_1 = A_2 [C]$,
 - ▶ $M_1 = M_2 : A [C]$.
- ▶ **Idea**: hypothetical judgement $A_1 = A_2 [C]$ holds if every categorical instance $A_1\rho = A_2\rho$ holds for $\rho : C$.
- ▶ For C valid , $\rho_1 = \rho_2 : C$, $A_1 = A_2 [C]$ and $M_1 = M_2 : A [C]$ are PERs.
- ▶ Hypothetical judgements satisfy the rules of Type Theory.

The point for this talk: **environments mention infinitely many variables.**

Environments

- ▶ An environment, ρ , is a function $var \rightarrow term$.
- ▶ ρ_0 is the identity environment.
- ▶ Environments are applied as simultaneous substitutions: $M\rho$, $A\rho$.
- ▶ Write $(\rho, x=M)$ for the *update* of ρ , defined by

$$(\rho, x=M)(x) = M, \quad (\rho, x=M)(y) = \rho(y) \text{ if } y \neq x.$$

Environments are infinite, and mention every identifier ...
... can freshness logics handle that?

Contexts

As usual ...

$$C ::= \nabla \mid C, x:A \quad (\nabla \text{ is the empty context.})$$

- ▶ Write $x \in C$ if $x:A$ in C for some A .
- ▶ In writing $C, x:A$ we assume $x \notin C$.

Equal environments

Inductively define a judgement of form $\rho_1 = \rho_2 : C$:

$$\overline{\rho_1 = \rho_2 : \nabla}$$

$$\frac{\rho_1 = \rho_2 : C \quad A\rho_1 \in \mathbf{Type} \quad \rho_1 x = \rho_2 x : \overline{A\rho_1}}{\rho_1 = \rho_2 : C, x:A}$$

Write $\rho : C$ for $\rho = \rho : C$.

- ▶ It may look like like finite environments would do ...
- ▶ ... but some delicate lemmas follow that I don't know how to prove with any finite presentation.

Hypothetical judgements defined

Simultaneously define three judgement forms:

validity

$$\frac{}{\nabla \text{ valid}} \quad \frac{x \notin C \quad A = A[C]}{C, x:A \text{ valid}}$$

type equality (write $A \text{ type } [C]$ for $A = A[C]$)

$$\frac{C \text{ valid} \quad \forall \rho_1, \rho_2 . \rho_1 = \rho_2 : C \implies A_1 \rho_1 = A_2 \rho_2}{A_1 = A_2 [C]}$$

object equality in a type (write $M : A [C]$ for $M = M : A [C]$)

$$\frac{A \text{ type } [C] \quad \forall \rho_1, \rho_2 . \rho_1 = \rho_2 : C \implies M_1 \rho_1 = M_2 \rho_2 : \overline{A} \rho_1}{M_1 = M_2 : A [C]}$$

Some delicate lemmas

Lemma (Hypothetical judgements make sense)

Let C be valid . The following relations are pers:

$$\begin{aligned} \rho_1, \rho_2 &\mapsto \rho_1 = \rho_2 : C, \\ A_1, A_2 &\mapsto A_1 = A_2 [C], \\ M_1, M_2 &\mapsto M_1 = M_2 : A [C] \end{aligned}$$

Lemma (Equality of environments only depends on context)

- ▶ *Let $\rho_1 = \rho_2 : C$ and $y \notin C$. For any M , $\rho_1 = (\rho_2, y=M) : C$.*
- ▶ *Let C be valid , $\rho_1 = \rho_2 : C$ and $y \notin C$. For any M and N , $(\rho_1, y=M) = (\rho_2, y=N) : C$.*

The proofs use the infinite representation of environments.

Infinite structures

- ▶ **Böhm trees** is another example of infinite structures that are used in practice (suggested by Barendregt).

Outline

Needed: A challenge problem set

Approaches to Formalizing Binding

- First Order Representations

- Higher Order Abstract Syntax (HOAS)

- Weak HOAS

- Logics with “freshness”

Some Naturally Occurring Formalisation Problems

- Eigenvariables: Freshness of globally bound variables

- Simultaneous Substitution for Structural Recursion

- Infinite Structures

Summary

Summary

- ▶ **Formal reasoning about binding is still difficult in practice.**
- ▶ Many technical approaches are proposed in the literature ...
 - ▶ (far fewer have usable implementations)
- ▶ ... but does any proposed approach support all the (correct) definitional and reasoning styles used informally?
- ▶ **Please tell me about other problematic examples you have encountered.**
- ▶ I look forward to continued technical progress in this area
 - ▶ especially with nominal approaches.