Proofs with Binders

Working on the POPLMark Challenge

A 5-month internship at the University of Pennsylvania
with Benjamin Pierce and Stephanie Weirich

Arthur Charguéraud

INRIA-Rocquencourt 14 Sept. 2006

Certification

Software → Compiler + Prover → Hardware

some day... | soon | already

Big Formalizations

– Coq in Coq

Bruno Barras and Benjamin Werner, 1996

– SML in Twelf

Karl Crary, Daniel Lee et Robert Harper, 2006

– C-light in Coq

Xavier Leroy, 2006

POPLMark: Challenge

Mechanized Metatheory for the Masses:
The POPLMark Challenge

By B. Aydemier, A. Bohannon, M. Fairbairn, N. Foster, B. Pierce, P. Sewell, D. Vytiniotis, G. Washburn, S. Weirich, S. Zdancewic (Mar.05)

- To formalize results from their POPL papers

- A set of benchmarks: Metatheory of System-$F_{<:}$

- Basis for comparing technologies and techniques

POPLMark: Rules

The formalization should:

1) be clearly adequate,

2) look like the paper version,

3) use general techniques,

4) have a reasonable cost,

5) use a transparent and
   accessible technology.

POPLMark: Contents

Preservation and Progress for System-F$_{<:}$

POPLMark Part 1:

Properties of subtyping

POPLMark Part 2:

Rest of the formalization

[simplified]

Our Challenge B

Properties of subtyping, including preservation by type substitution

Our Challenge A

Preservation and progress for simply typed λ-calculus

A) Simply Typed λ-calculus

$$
\begin{array}{lcllllll}
T & := & A & | & T_1 \rightarrow T_2 & & \\
t & := & x & | & (t_1\ t_2) & | & \lambda x{:}T.\ t_1
\end{array}
$$

$$
\frac{(x : T) \in E}{E \vdash x\ :\ T}\ \textsc{T-Var}
\qquad
\frac{E \vdash t_1\ :\ S \rightarrow T \qquad E \vdash t_2\ :\ S}{E \vdash (t_1\ t_2)\ :\ T}\ \textsc{T-App}
$$

$$
\frac{x \# E \qquad E, x : S \vdash t_1\ :\ T}{E \vdash (\lambda x{:}S.\ t_1)\ :\ S \rightarrow T}\ \textsc{T-Abs}
$$

Preservation: $t \longmapsto t' \quad \wedge \quad E \vdash t\ :\ T \quad \Rightarrow \quad E \vdash t'\ :\ T$

Progress: $\varnothing \vdash t\ :\ T \quad \Rightarrow \quad \left[\, t \text{ is a value} \quad \vee \quad \exists t',\ t \longmapsto t' \,\right]$

B) Subtyping in System-$F_{<:}$

$$T \quad := \quad \mathrm{Top} \quad | \quad X \quad | \quad T_1 \to T_2 \quad | \quad \forall X{<:}T_1.\, T_2$$

$$\frac{}{E \vdash S <: \mathrm{Top}} \ \text{SA-TOP} \qquad \frac{E \vdash T_1 <: S_1 \qquad E \vdash S_2 <: T_2}{E \vdash (S_1 \to S_2) <: (T_1 \to T_2)} \ \text{SA-ARROW}$$

$$\frac{}{E \vdash X <: X} \ \text{SA-REFL-TVAR} \qquad \frac{(X{<:}U) \in E \qquad E \vdash U <: T}{E \vdash X <: T} \ \text{SA-TRANS-TVAR}$$

$$\frac{E \vdash T_1 <: S_1 \qquad X \# E \qquad (E,\, X{<:}T_1) \vdash S_2 <: T_2}{E \vdash (\forall X{<:}S_1.\, S_2) <: (\forall X{<:}T_1.\, T_2)} \ \text{SA-ALL}$$

Reflexivity: $\boxed{E \vdash T <: T}$

Transtitivity: $\boxed{E \vdash S <: Q \quad \wedge \quad E \vdash Q <: T \quad \Rightarrow \quad E \vdash S <: T}$

Preservation by type substitution:

$$E,\, Z{<:}Q, F \vdash S <: T \quad \wedge \quad E \vdash P <: Q$$
$$\Rightarrow \quad E, [Z \to P]\, F \vdash [Z \to P]\, S <: [Z \to P]\, T$$

Concrete versus Higher-Order

Deep embedding is attractive but:

– adequacy is often not so obvious,

– proofs do not follow informal practice,

– logic used have limited expressiveness.

Previous Work

Yr	Author	Formalization	Prover	Encoding
85	Natarajan Shankar	Church-Rosser in λ	Boyer-Moore	de-Bruijn
93	Thorsten Altenkirch	System-F	LEGO	de-Bruijn
93	J.McKinna, R.Pollack	Pure Type Systems	LEGO	de-Bruijn
94	Gérard Huet	Residual Theory in λ	Coq	de-Bruijn
95	Ole Rasmussen	Church-Rosser in λ	Isabelle/ZF	de-Bruijn
96	Tobias Nipkow	Church-Rosser in λ	Isabelle/HOL	de-Bruijn
96	B.Barras, B.Werner	Kernel of Coq	Coq	de-Bruijn
97	J.McKinna, R.Pollack	λ-calculus & types	LEGO	names
01	Vestergaard, Brotherston	Church-Rosser in λ	Isabelle/HOL	names
01	J.Ford, I.Mason	Church-Rosser in λ	PVS	names
01	Peter Homeier	Church-Rosser in λ	HOL	names

POPLMark: Submissions

Author	Part 1	Part 2	Prover	Encoding
Stephan Berghofer	Y	Y	Isabelle	de-Bruijn indices
Ashley, Crary, Harper	Y	Y	Twelf	higher-order
Jérome Vouillon	Y	Y	Coq	de-Bruijn indices
Hongwei Xi		Y	ATS/LF	higher-order
Jevgenijs Sallinens	Y		Coq	de-Bruijn indices
Xavier Leroy	Y		Coq	locally nameless
Aaron Stump	Y		Coq	names / levels
Christian Urban	Y		Isabelle	nominal
Hirschowitz, Maggesi	Y		Coq	de-Bruijn (nested)
Adam Chlipala	Y		Coq	locally nameless
Arthur Charguéraud	Y		Coq	locally nameless

Contribution

– Gather and compare techniques for such formalizations in one paper.

– Provide examples of formalizations in Coq which are rather simple and intuitive.

Techniques

Plan

1) Bindings

- representation of bound and free variables,
- implementation of substitution and β-reduction.

2) Well-formation

- well-formation of terms, induction on terms,
- well-formation in typing/subtyping relations.

3) Environments

- algorithmic and logical views on environments,
- properties of well-formed environments.

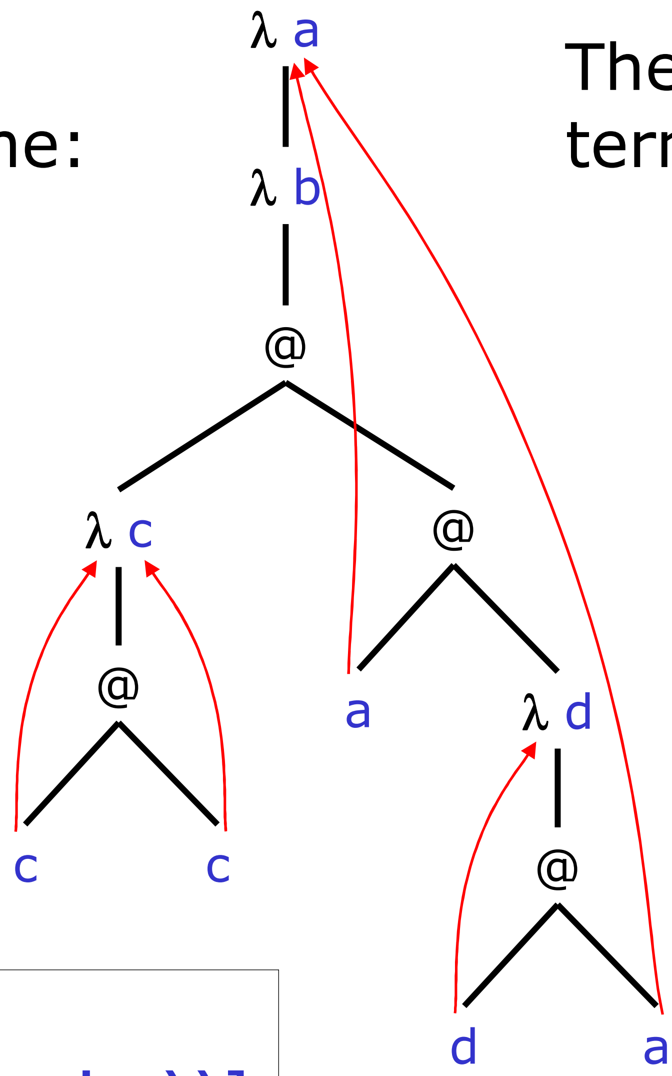
4) Quantification of names

- how to introduce names for typing abstractions.

1) Bindings

λ-term with names

Each abstraction introduces a name:



Then α-equivalent terms are identified.

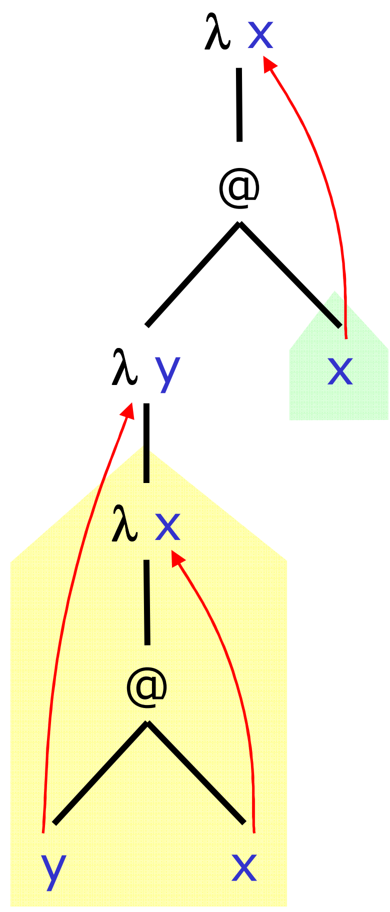
λa.λb.
[(λc. c c) (a (λd. d a))]

β-reduction with names

```
   (λz. z z) (λy. λx. y x)
-> (λy. λx. y x) (λy. λx. y x)
-> λx. [(λy. λx. y x) x]
```

α-conversion required before substituting

```
-> λx. [(λy. λz. y z) x]
-> λx. [λz. x z]
```



Handling α-conversion

With quotient

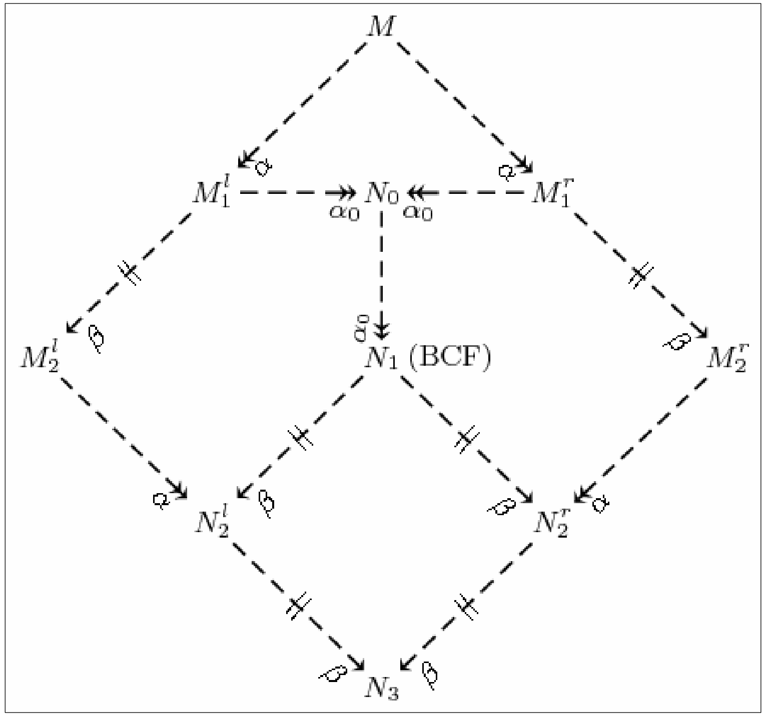
[Homeier, 2001] [Ford & Mason, 2001]

Corollary 24 (Respectfulness of substitution).

$$t_1 \equiv_\alpha t_2 \ \wedge\ (\forall x.\ x \in \mathsf{FV}_1\ t_1 \ \Rightarrow\ (x \lhd_1^v s_1) \equiv_\alpha (x \lhd_1^v s_2)) \ \Rightarrow\ (t_1 \lhd_1 s_1) \equiv_\alpha (t_2 \lhd_1 s_2)$$

Without quotient

[Verstergaard & Brotherston, 2001]



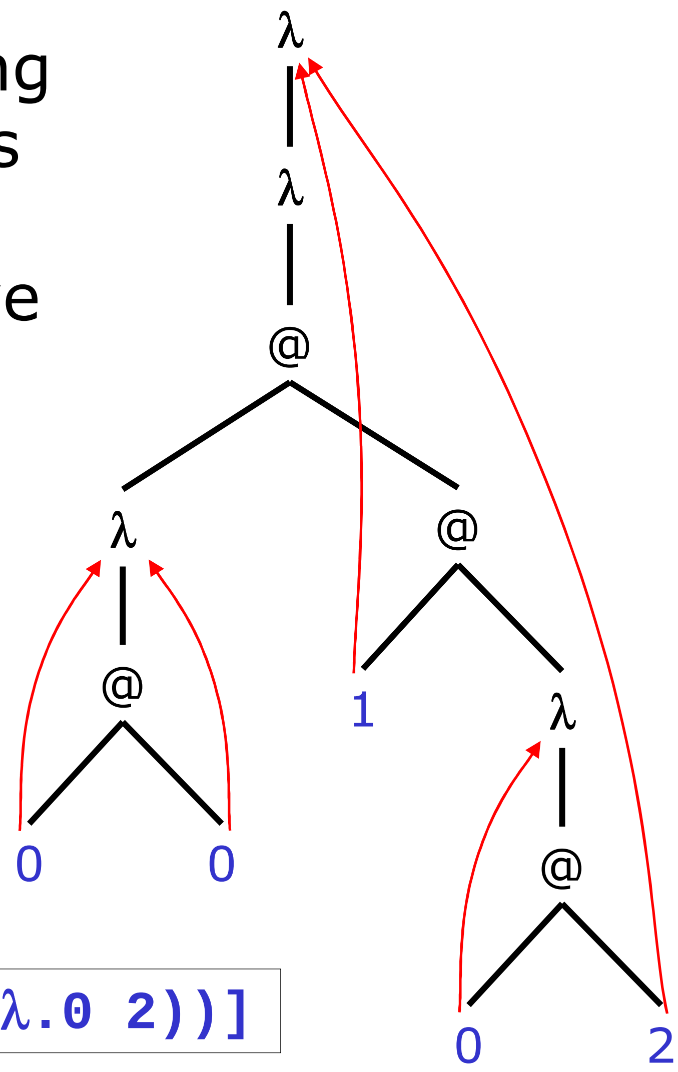
Without quotient nor identification

[McKinna & Pollack, 1997]

$$A \downarrow M \ \Rightarrow\ (A \downarrow N \ \Leftrightarrow\ M \overset{\alpha}{\sim} N).$$

λ-term with de-Bruijn indices

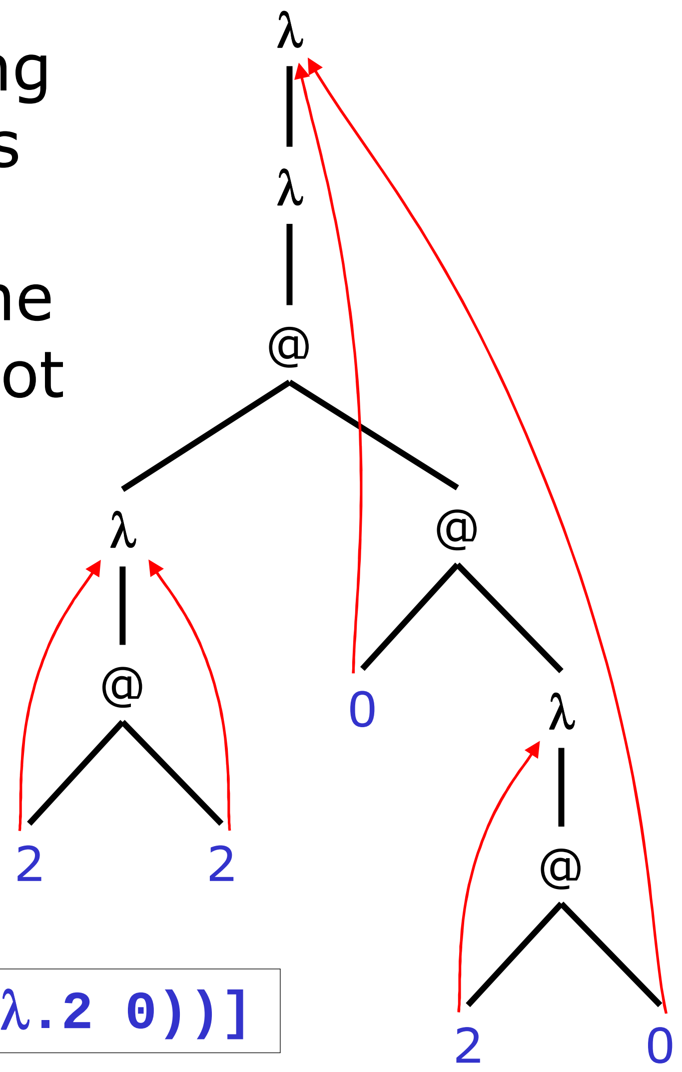
A variable bearing an index *k* points towards the *k*ith abstraction above that variable:



`λ.λ.[(λ.0 0) (1 (λ.0 2))]`

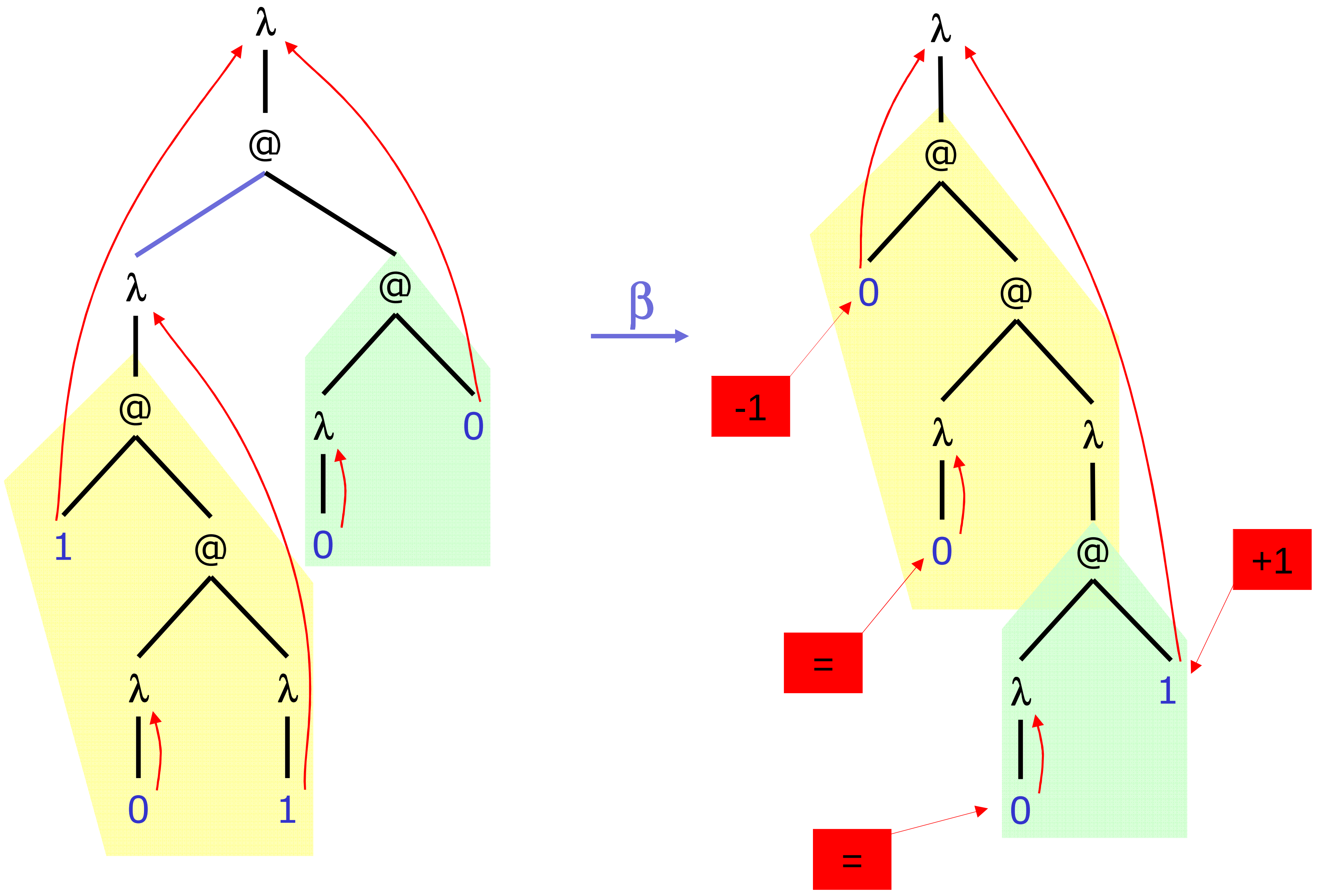
λ-term with de-Bruijn levels

A variable bearing an index *k* points towards the k[ith] abstraction on the path from the root to that variable:



`λ.λ.[(λ.2 2) (0 (λ.2 0))]`

β-reduction with de-Bruijn indices



shift and *subst*

Properties of shifting and substitution: [Berghofer, 2005]

$$
\begin{array}{l}
i \leq j \Longrightarrow j \leq i + m \Longrightarrow \uparrow_\tau n\ j\ (\uparrow_\tau m\ i\ T) = \uparrow_\tau (m + n)\ i\ T \\
i + m \leq j \Longrightarrow \uparrow_\tau n\ j\ (\uparrow_\tau m\ i\ T) = \uparrow_\tau m\ i\ (\uparrow_\tau n\ (j - m)\ T) \\
k \leq k' \Longrightarrow k' < k + n \Longrightarrow \uparrow_\tau n\ k\ T[k' \mapsto_\tau U]_\tau = \uparrow_\tau (n - 1)\ k\ T \\
k \leq k' \Longrightarrow \uparrow_\tau n\ k\ (T[k' \mapsto_\tau U]_\tau) = \uparrow_\tau n\ k\ T[k' + n \mapsto_\tau U]_\tau \\
k' < k \Longrightarrow \uparrow_\tau n\ k\ (T[k' \mapsto_\tau U]_\tau) = \uparrow_\tau n\ (k + 1)\ T[k' \mapsto_\tau \uparrow_\tau n\ (k - k')\ U]_\tau \\
k \leq k' \Longrightarrow \uparrow_\tau n\ k'\ (T[k \mapsto_\tau Top]_\tau) = \uparrow_\tau n\ (Suc\ k')\ T[k \mapsto_\tau Top]_\tau \\
k \leq k' \Longrightarrow k' \leq k + n \Longrightarrow \uparrow n'\ k'\ (\uparrow n\ k\ t) = \uparrow (n + n')\ k\ t \\
i \leq j \Longrightarrow T[Suc\ j \mapsto_\tau V]_\tau[i \mapsto_\tau U[j - i \mapsto_\tau V]_\tau]_\tau = T[i \mapsto_\tau U]_\tau[j \mapsto_\tau V]_\tau
\end{array}
$$

Substitution in simply typed lambda-calculus:

$$(E, :U) \vdash t : T \quad \wedge \quad E \vdash u : U \quad \Rightarrow \quad E \vdash \downarrow^0_u t : T$$

Weakening in System-$F_{<:}$

$$\Gamma \vdash t : T \Longrightarrow \Delta\ @\ \Gamma \vdash_{wf} \Longrightarrow \Delta\ @\ \Gamma \vdash \uparrow \|\Delta\|\ 0\ t : \uparrow_\tau \|\Delta\|\ 0\ T$$

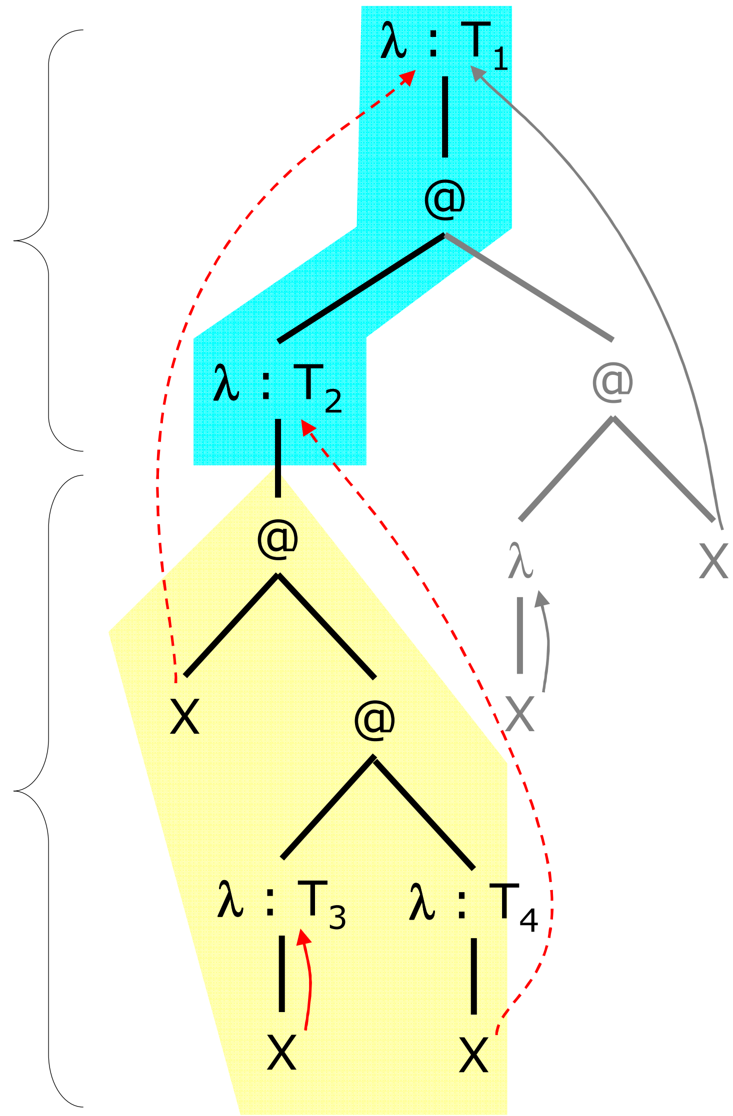
Bound and Free Variables

environment E

typing judgment

$$\boldsymbol{E \ |\!- \ t \ : \ T}$$

term t



Distinguishing Bound and Free

Example with the locally nameless representation.

Substitution for a bound variable (de-Bruijn index):

$$
\begin{array}{lcl}
\{k \to u\}\,\lceil i \rceil & \equiv & \text{if } i = k \text{ then } u \text{ else } \lceil i \rceil \\
\{k \to u\}\,\lfloor x \rfloor & \equiv & \lfloor x \rfloor \\
\{k \to u\}\,(t_1\ \ t_2) & \equiv & ((\{k \to u\}\,t_1)\ \ (\{k \to u\}\,t_2)) \\
\{k \to u\}\,(\lambda{:}T.\,t_1) & \equiv & \lambda{:}T.\,(\{(k+1) \to u\}\,t_1)
\end{array}
$$

Substitution for a free variable (name):

$$
\begin{array}{lcl}
[z \to u]\,\lceil i \rceil & \equiv & \lceil i \rceil \\
[z \to u]\,\lfloor x \rfloor & \equiv & \text{if } x = z \text{ then } u \text{ else } \lfloor x \rfloor \\
[z \to u]\,(t_1\ \ t_2) & \equiv & (([z \to u]\,t_1)\ \ ([z \to u]\,t_2)) \\
[z \to u]\,(\lambda{:}T.\,t_1) & \equiv & \lambda{:}T.\,([z \to u]\,t_1)
\end{array}
$$

β-reduction in Locally Nameless

$$\frac{}{((\lambda{:}S.\ t_1)\ \ t_2) \longmapsto {t_1}^{t_2}}\ \text{RED-BETA}$$

$$\frac{t_1 \longmapsto t_1'}{(t_1\ \ t_2) \longmapsto (t_1'\ \ t_2)}\ \text{RED-APP-1} \qquad \frac{t_2 \longmapsto t_2'}{(t_1\ \ t_2) \longmapsto (t_1\ \ t_2')}\ \text{RED-APP-2}$$

$$\frac{\forall\, x \,\#\, t_1,\quad {t_1}^{x} \longmapsto {t_1'}^{x}}{(\lambda{:}S.\ t_1) \longmapsto (\lambda{:}S.\ t_1')}\ \text{RED-ABS}$$

${t_1}^{t_2}$ stands for $\{0 \to t_2\}\, t_1$.

Most Used Representations

Representation	Grammar
Mixed names	$x \mid (t_1\ t_2) \mid \lambda x{:}S.\ t_1$
Mixed indices	$i \mid (t_1\ t_2) \mid \lambda{:}S.\ t_1$
Distinct names	$\lceil y \rceil \mid \lfloor x \rfloor \mid (t_1\ t_2) \mid \lambda y{:}S.\ t_1$
Indices / levels	$\lceil i \rceil \mid \lfloor k \rfloor \mid (t_1\ t_2) \mid \lambda x{:}S.\ t_1$
Locally nameless	$\lceil i \rceil \mid \lfloor x \rfloor \mid \quad \mid (t_1\ t_2) \mid \lambda{:}S.\ t_1$

Presentations for T-abs

Standard:

$$\frac{Q(x) \qquad (E,\, x : S) \vdash t_1 \;:\; T}{E \vdash (\lambda x{:}S.\, t_1) \;:\; S \to T}$$

Mixed names:

$$\frac{Q(x) \qquad (E,\, x : S) \vdash [y \to x]\,t_1 \;:\; T}{E \vdash (\lambda y{:}S.\, t_1) \;:\; S \to T}$$

Distinct names:

$$\frac{Q(x) \qquad (E,\, x : S) \vdash [y \to \lfloor x \rfloor]\,t_1 \;:\; T}{E \vdash (\lambda y{:}S.\, t_1) \;:\; S \to T}$$

Locally nameless:

$$\frac{Q(x) \qquad (E,\, x : S) \vdash {t_1}^{x} \;:\; T}{E \vdash (\lambda{:}S.\, t_1) \;:\; S \to T}$$

Indices / levels:

$$\frac{(E,\, : S) \vdash {t_1}^{|E|} \;:\; T}{E \vdash (\lambda{:}S.\, t_1) \;:\; S \to T}$$

Mixed indices:

$$\frac{(E,\, : S) \vdash t_1 \;:\; T}{E \vdash (\lambda{:}S.\, t_1) \;:\; S \to T}$$

where Q(x) to be choosen among: $\exists x \# E$ or $\forall x \# E$ or $\forall x \notin L$.

Locally Nameless: Bibliography

1972: N.G. de-Bruijn
A Lambda Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem.

1989: G. Huet
The Constructive Engine

1993: A. Gordon
A Mechanisation of Name-carrying Syntax up to Alpha-conversion

1995: R. Pollack
Closure under Alpha-conversion

2004: C. McBride and J. McKinna,
I am not a number: I am a free variable

2005-2006: X. Leroy, A. Chipala, A. Charguéraud,
Independent solutions to the POPLMark Challenge.

2) Well-formation

Need for Well-formation

Examples of terms not well-formed:

In the environment **`z:T`**, the term λ**`x.z y`** is ill-formed

In the environment **`[T;U]`**, the term λ.**`5`** is ill-formed.

Ill-formed terms need to be ruled out:

e.g. reflexitivity of subtyping **`E |- T <: T`**

does not hold if **`T`** is a variable not defined in **`E`**.

Well-formed terms

–With recursive functions:

- for names: *all variables in the term belong to a domain*
- for indices: *all indices in the term are smaller than a natural*

– With dependent types

`t : term n` instead of **`t : term`** $\wedge$ **`wf n t`**

– With inductive relation:

$$\frac{(x:U) \in E}{E \vdash \lfloor x \rfloor \ \mathit{wf}} \ \text{WF-FVAR} \qquad \frac{E \vdash t_1 \ \mathit{wf} \qquad E \vdash t_2 \ \mathit{wf}}{E \vdash (t_1 \ t_2) \ \mathit{wf}} \ \text{WF-APP}$$

$$\frac{\forall x \notin L, \ (E, x:T) \vdash {t_1}^x \ \mathit{wf}}{E \vdash (\lambda{:}T.\, t_1) \ \mathit{wf}} \ \text{WF-ABS}$$

Induction on Well-formed Terms

Informal statement: $\boxed{E \vdash T <: T}$

Informal proof:

Trivial by induction on T.

Formal statement: $\boxed{\vdash E\ ok \quad \wedge \quad E \vdash T\ wf \quad \Rightarrow \quad E \vdash T <: T}$

Formal proof:

```
Lemma sub_reflexivity : forall E T,
  ok E  ->  E wf T  ->  E |- T <: T.
intros. induction H0; eauto.
Qed.
```

Well-formation in Relations

$$\frac{}{\boxed{E} \vdash \boxed{S} <: \mathrm{Top}}\ \textsc{SA-Top} \qquad \frac{E \vdash T_1 <: S_1 \qquad E \vdash S_2 <: T_2}{E \vdash (S_1 \to S_2) <: (T_1 \to T_2)}\ \textsc{SA-Arrow}$$

$$\frac{}{\boxed{E} \vdash \boxed{X} <: \boxed{X}}\ \textsc{SA-Refl-TVar} \qquad \frac{(X{<:}U) \in E \qquad E \vdash U <: T}{E \vdash X <: T}\ \textsc{SA-Trans-TVar}$$

$$\frac{E \vdash T_1 <: S_1 \qquad X \# E \qquad (E, X{<:}T_1) \vdash S_2 <: T_2}{E \vdash (\forall X{<:}S_1.\, S_2) <: (\forall X{<:}T_1.\, T_2)}\ \textsc{SA-All}$$

Where to store the well-formation of arguments?

- At all nodes
- At all leaves
- At the root

3) Environments

Environments as Sets

Weakening Preserves Typing

From: $\boxed{E \vdash S <: T \quad \Rightarrow \quad E, F \vdash S <: T}$

To: $\boxed{E \vdash S <: T \quad \wedge \quad E \subset F \quad \Rightarrow \quad F \vdash S <: T}$

With: $\boxed{E \subset F \quad \equiv \quad \forall x\, T, \quad (x : T) \in E \quad \Rightarrow \quad (x : T) \in F}$

Substitution Preserves Typing

From: $\boxed{E, z : U, F \vdash t \, : \, T \quad \wedge \quad E \vdash u \, : \, U \quad \Rightarrow \quad E, F \vdash [z \rightarrow u]t \, : \, T}$

To:

$$E \vdash t \, : \, T \quad \wedge \quad F \vdash u \, : \, U \quad \wedge$$
$$(z : U) \in E \quad \wedge \quad E \setminus z \subset F \quad \Rightarrow \quad F \vdash [z \rightarrow u]t \, : \, T$$

With: $\boxed{E \setminus z \subset F \quad \equiv \quad \forall x\, T, \quad (x : T) \in E \quad \wedge \quad x \neq z \quad \Rightarrow \quad (x : T) \in F}$

Views on Environments

view	structure	lookup	weaken
function	`list`	`lookup x` Γ `= Some U`	Γ, Δ
relation	`set`	`(x:U)` $\in$ `E`	`(x:U)`$\in$ `E` $\Rightarrow$ `(x:U)`$\in$ `F`

E $\subset$ F

view	substitution	type substitution
function	Γ`,z:T,`Δ `to` Γ`,`Δ	Γ`,Z<:Q,`Δ `to` Γ`,[Z->P]`Δ
relation	`(x:U)`$\in$ `E` $\wedge$ `x` $\neq$ `z` $\Rightarrow$ `(x:U)`$\in$ `F`	`(X <: U)`$\in$ `E` $\wedge$ `X` $\neq$ `Z` $\Rightarrow$ `(X <:[Z->P]U)`$\in$ `F`

E\z $\subset$ F

[Z->P]E $\subset$ F

Freshness, Closed Terms

	`x # E`	`x # t`	`t closed term`
function	`x` $\notin$ `dom(E)`	`x` $\notin$ `FV(t)`	`FV(t) =` $\varnothing$
inductive relation	`not_in_env x E`	`not_in_term x t`	`closed t`
property	`forall U,` `(x:U)` $\notin$ `E`	`E |- t wf` $\wedge$ `x # E`	$\varnothing$ `|- t wf`

4) Quantification

Quantification of names

$$\frac{\mathrm{Q}(x) \qquad (E, x : S) \vdash {t_1}^x \; : \; T}{E \vdash (\lambda{:}S.\, t_1) \; : \; S \rightarrow T}$$

Q(x) =	∃ x # E	∀ x # E	∀ x ∉ L

Weakening: $E \vdash t \; : \; T \quad \Rightarrow \quad E, F \vdash t \; : \; T$

```
from    Q(x)  (E, x:S) |- t1^x : T
to      Q(x)  (E, x:S, F) |- t1^x : T
```

```
∃ x # (E, x:S)    to    ∃ x # (E, x:S, F)
∀ x # (E, x:S)    to    ∀ x # (E, x:S, F)
∀ x ∉ L           to    ∀ x ∉ L'
```

Comparing Quantification

Weakening	$E \vdash t : T \quad \Rightarrow \quad E, F \vdash t : T$
Substitution	$E, z : U, F \vdash t : T \quad \wedge \quad E \vdash u : U \quad \Rightarrow \quad E, F \vdash [z \rightarrow u]t : T$
Transitivity	$E \vdash S <: Q \quad \wedge \quad E \vdash Q <: T \quad \Rightarrow \quad E \vdash S <: T$

Q(x) =	∃ x # E	∀ x # E	∀ x ∉ L
weakening	problem when x ∈ dom F	ok	take dom F
substitution	take x	problem when x = z	take L ∪ {z}
transitivity	problem when x ≠ x′	ok	take L ∪ L′

Proofs in Coq

Summary of Our Choices

A locally nameless solution

- bound variables are represented with de-Bruijn indices,
- free variables are represented with names,
- two simple substitutions, one for indices and one for names,
- well-formation defined inductively, gives induction principle.

With environments viewed as sets

- belonging relation **`(x:U)`** $\in$ **`E`**
- weakening as set inclusion **`E`** $\subset$ **`F`**
- substitution as **`E\z`** $\subset$ **`F`** or **`[Z->P]E`** $\subset$ **`F`**

And typing/subtyping relations defined

- with well-formation of all arguments at each node,
- quantifying names of free variables over a cofinite set.

Weakening on Subtyping

Informal:

$$E \vdash S <: T \quad \Rightarrow \quad E, F \vdash S <: T$$

Proof by induction on the subtyping derivation, using the reordering lemma for case SA-all.

Formalizable:

$$E \vdash S <: T \quad \wedge \quad E \subset F \quad \wedge \quad \vdash F \; ok \quad \Rightarrow \quad F \vdash S <: T$$

Proof by induction on the subtyping derivation, using extension of inclusion lemma in case SA-all and quantifying not among L ∪ dom(F).

Formal:

```
Lemma sub_extension : forall E S T, E |- S <: T
    -> forall F, E inc F -> ok F -> F |- S <: T.
intros E S T H. induction H; intros; auto**.
apply_SA_all X (L ++ dom F). use extends_push.
```

« Formalizable » Presentation

Informal Presentation → Formalizable Presentation → Formal Proofs

A better way to present the proofs on paper?

- same structure and key ideas as informally,
- definitions and statement of lemmas change slightly,
- can be written by hand (not too heavy),
- can be translated almost word-to-word into Coq.

Importance of Proof-search

Formal proofs like this contain many arguments.

Proof-search can help when:

– only easy steps of reasoning are involved,
– the statements combine well together,
– even if the chain of reasoning is rather long.

Proof-search will not help for:

– invoking key lemmas,
– performing inductions or case analysis,
– dealing with equalities.

Transitivity of Subtyping

```
Definition sub_trans_prop E Q (WQ : E wf Q) := forall F S T,
 E inc F  ->   F |- S <: Q  ->  F |- Q <: T  ->  F |- S <: T.


Lemma sub_transitivity :
 forall E Q (WQ : E wf Q), sub_trans_prop WQ.
intros. unfold sub_trans_prop. generalize_equality Q Q'.
induction WQ; intros Q' EQ F S T EincF SsubQ QsubT;
induction SsubQ; try discriminate; try injection EQ; intros;
inversion QsubT; subst; intuition eauto.
  (* Case SA-arrow *)
apply SA_arrow. auto. apply* IHWQ1. apply* IHWQ2.
  (* Case SA-all *)
apply_SA_all X ((dom E0) ++ L ++ L0 ++ L1). apply* H0.
asserts* WQ1 (E0 wf T1). apply* (sub_narrowing (WQ := WQ1)).
Qed.
```

Final Theorems for Subtyping

- SUBTYPING-REFLEXIVITY:

$$\vdash E\ ok \quad \wedge \quad E \vdash T\ wf \quad \Rightarrow \quad E \vdash T <: T$$

- SUBTYPING-WEAKENING:

$$E \vdash S <: T \quad \wedge \quad E \subset F \quad \wedge \quad \vdash F\ ok \quad \Rightarrow \quad F \vdash S <: T$$

- SUBTYPING-NARROWING:

$$E, Z{<:}Q \vdash S <: T \quad \wedge \quad E \vdash P <: Q \quad \Rightarrow \quad E, Z{<:}P \vdash S <: T$$

- SUBTYPING-TRANSITIVITY:

$$E \vdash S <: Q \quad \wedge \quad E \vdash Q <: T \quad \Rightarrow \quad E \vdash S <: T$$

- SUBTYPING-SUBSTITUTION:

$$(E, Z{<:}Q) \vdash S <: T \quad \wedge \quad E \vdash P <: Q \quad \Rightarrow \quad E \vdash [Z \rightarrow P]\,S <: [Z \rightarrow P]\,T$$

Statistics on our Coq Scripts

	Simply typed λ-calculus		Properties of subtyping
Definitions	8		9
Axioms	0		0
Lemmas	26	18 in common	34
Theorems	2		5
Lines of proofs	63		104
Number of tactics	202		279
...in main proofs	44		80
Non-empty lines	289		397

Complexity of solutions in Coq

Number of tactics invoked (not counting *trivial*, *assumption*, and *auto*) in solutions in Coq to part 1A of the POPLMark Challenge (formalization of the basic properties of subtyping), in chronological order. Column *Hints* gives the number of lemmas placed in the proof-search database.

Author	Steps	Hints	Representation
Jérome Vouillon	431	0	de-Bruijn indices
Aaron Stump	1147	0	names / levels
Xavier Leroy	630	3	locally nameless
Hirschowitz, Maggesi	1615	5	de-Bruijn (nested)
Adam Chlipala	342	70	locally nameless
Arthur Charguéraud	233	12	locally nameless

Conclusions

A Very Positive Experience

- A project with a clear and precise goal.
- Motivating to see progress.
- 5 months gives time for in-depth search.
- A very good environment of work.
- I learned a lot about many things.

Improvements for Coq

– It is already an impressive tool, and we never felt limited by Coq.

– The structure of the proofs should be stored in a better way, so as to make proofs more robust.

– Proof-search, once successful, should store the main steps followed, to improve efficiency.

Future Work

– Complete the solution to cover the rest of the POPLMark Challenge.

– Extend the results to more evolved languages, with support for many constructions.

– Extend the results to more complex typing systems (e.g. Calculus of Constructions).

Thanks !