A weak HOAS approach to the POPLmark Challenge

Alberto Ciaffaglione    Ivan Scagnetto

Università di Udine, Italia
Dipartimento di Matematica e Informatica
{alberto.ciaffaglione, ivan.scagnetto}@uniud.it

LSFA 2012 - 7th Workshop on Logical and Semantic Frameworks, with Applications
Rio de Janeiro, Brasil - September 29-30, 2012
Outline

1. Introduction
2. System $F_{<}$
3. Formalization
4. Conclusion
The POPLmark Challenge

Formal proofs about programming language metatheory/semantics can be long and tedious due to the management of the details. Small mistakes or missed subtle cases can be harmful. Automated proof assistants may help, with potential benefits such as reusing the work, keeping definitions and proofs consistent, and ensuring a firm relationship between theory and implementation. A framework and a set of problems for measuring the progress framework System F\texttt{<:} polymorphic (second-order) \lambda-calculus, which includes variable binding, complex recursion and induction, definition and proof reuse, and experimentation of generated sample programs.
The POPLmark Challenge

- Formal proofs about **programming language** metatheory/semantics
  - long and tedious
  - management of the details
  - small mistakes or missed subtle cases harmful
The POPLmark Challenge

- Formal proofs about programming language metatheory/semantics
  - long and tedious
  - management of the details
  - small mistakes or missed subtle cases harmful
- Automated proof assistants may help, with potential benefits
  - reusing the work
  - keeping definitions and proofs consistent
  - ensuring a firm relationship between theory and implementation
The POPLmark Challenge

- Formal proofs about *programming language* metatheory/semantics
  - long and tedious
  - management of the details
  - small mistakes or missed subtle cases harmful
- Automated *proof assistants* may help, with potential benefits
  - reusing the work
  - keeping definitions and proofs consistent
  - ensuring a firm relationship between theory and implementation
- **A framework and a set of problems for measuring the progress**
  - **framework** System F_{<:}: polymorphic (second-order) $\lambda$-calculus
  - **problems** variable binding, complex recursion and induction, definition and proof reuse, experimentation of generated sample programs
Our contribute
Our contribute

- System $F_{\lt:}$’s type language (variable binding, complex induction)
Our contribute

- System $\text{F}_{\prec}$’s **type language** (variable binding, complex induction)
- logical frameworks based on type theory (LFs):
  the *Calculus of Inductive Constructions*, and its *Coq* implementation
Our contribute

- System $\text{F}_{<:}$'s type language (variable binding, complex induction)
- logical frameworks based on type theory (LFs):
  the *Calculus of Inductive Constructions*, and its *Coq* implementation
- management of inductively-defined structures with binders
Our contribute

- System $F_{<\cdot}$’s type language (variable binding, complex induction)
- logical frameworks based on type theory (LFs):
  - the *Calculus of Inductive Constructions*, and its *Coq* implementation
- management of inductively-defined structures with binders
  - Higher-Order Abstract Syntax (HOAS)
Our contribute

- System $F_{<:}$'s type language (variable binding, complex induction)
- logical frameworks based on type theory (LFs):
  the *Calculus of Inductive Constructions*, and its *Coq* implementation
- management of inductively-defined structures with binders
  - Higher-Order Abstract Syntax (HOAS)
  - weak HOAS (second-order term constructors that take as arguments functions over a parametric type of variables)
Our contribute

- System $F_{\ll}:'$s type language (variable binding, complex induction)
- logical frameworks based on type theory (LFs):
  - the *Calculus of Inductive Constructions*, and its *Coq* implementation
- management of inductively-defined structures with binders
  - Higher-Order Abstract Syntax (HOAS)
  - weak HOAS (second-order term constructors that take as arguments functions over a parametric type of variables)
  - the Theory of Contexts
- System $F_{<:}$’s type language (variable binding, complex induction)
- logical frameworks based on type theory (LFs):
  - the *Calculus of Inductive Constructions*, and its *Coq* implementation
- management of inductively-defined structures with binders
  - Higher-Order Abstract Syntax (HOAS)
  - weak HOAS (second-order term constructors that take as arguments functions over a parametric type of variables)
  - the Theory of Contexts
- we prove the first (of the three) task(s) of the Challenge: transitivity (and narrowing) of algorithmic subtyping
The (pure) type language
The (pure) type language

- Syntax of types:

\[
\text{Type} : \quad S, T ::= X \quad \text{type variable} \\
\quad \quad \text{Top} \quad \text{maximal type} \\
\quad \quad S \to T \quad \text{function type} \\
\quad \quad \forall X <: S. T \quad \text{universal type}
\]
The (pure) type language

- Syntax of types:

  \[
  \text{Type} : \quad S, T ::= \quad X \quad \text{type variable} \\
  \quad \text{Top} \quad \text{maximal type} \\
  \quad S \to T \quad \text{function type} \\
  \quad \forall X ::= S \cdot T \quad \text{universal type}
  \]

- Syntax of type environments:

  \[
  \text{Env} : \quad \Gamma, \Gamma' ::= \quad \emptyset \quad \text{empty type environment} \\
  \quad \Gamma', X ::= T \quad \text{type variable binding} \text{ (with scoping discipline)}
  \]
Algorithmic subtyping (for well-scoped types)
Algorithmic subtyping (for well-scoped types)

- Subtyping:
  \[
  \Gamma \vdash S <: Top \quad (\text{Top}) \quad \Gamma \vdash X <: X \quad (\text{Refl})
  \]
  \[
  \frac{X <: U \in \Gamma \quad \Gamma \vdash U <: T}{\Gamma \vdash X <: T} \quad (\text{Trans})
  \]
  \[
  \frac{\Gamma \vdash T_1 <: S_1 \quad \Gamma \vdash S_2 <: T_2}{\Gamma \vdash S_1 \rightarrow S_2 <: T_1 \rightarrow T_2} \quad (\text{Arr})
  \]
  \[
  \frac{\Gamma \vdash T_1 <: S_1 \quad \Gamma, X <: T_1 \vdash S_2 <: T_2}{\Gamma \vdash \forall X <: S_1, S_2 <: \forall X <: T_1, T_2} \quad (\text{All})
  \]
Algorithmic subtyping (for well-scoped types)

- Subtyping:
  
  \[ \Gamma \vdash S <: Top \quad (\text{Top}) \quad \Gamma \vdash X <: X \quad (\text{Refl}) \]

  \[ \begin{align*}
  \frac{X <: U \in \Gamma \quad \Gamma \vdash U <: T}{\Gamma \vdash X <: T} \quad (\text{Trans})
  
  \frac{\Gamma \vdash T_1 <: S_1 \quad \Gamma \vdash S_2 <: T_2}{\Gamma \vdash S_1 \to S_2 <: T_1 \to T_2} \quad (\text{Arr})
  
  \frac{\Gamma \vdash T_1 <: S_1 \quad \Gamma, X <: T_1 \vdash S_2 <: T_2}{\Gamma \vdash \forall X <: S_1. S_2 <: \forall X <: T_1. T_2} \quad (\text{All})
  \end{align*} \]

- Proposition 1 (Transitivity and Narrowing)
  
  \[ \Gamma \vdash S <: Q \land \Gamma \vdash Q <: T \Rightarrow \Gamma \vdash S <: T \]
  
  \[ \Gamma, X <: Q, \Delta \vdash M <: N \land \Gamma \vdash P <: Q \Rightarrow \Gamma, X <: P, \Delta \vdash M <: N \]

  **Proof:** By (outer) induction on \( Q \).
Transitivity and Narrowing

Transitivity: by inner induction on the derivation $\Gamma \vdash S < Q$. . .

$\Gamma \vdash Q_1 < S_1$. . .

$\Gamma, X < Q_1 \vdash S_2 < Q_2$. . .

$\Gamma \vdash \forall X < S_1. S_2 < \forall X < Q_1. Q_2$. . .

$\Gamma \vdash T_1 < Q_1$. . .

$\Gamma, X < T_1 \vdash Q_2 < T_2$. . .

$\Gamma \vdash \forall X < Q_1. Q_2 < \forall X < T_1. T_2$. . .

To conclude (via the (All) rule):

$\Gamma \vdash T_1 < S_1$ . . .

$\Gamma, X < T_1 \vdash S_2 < T_2$. (via narrowing: $Q_1$ structurally smaller than $Q_2$)

Narrowing: by inner induction on $\Gamma, X < Q, \Delta \vdash M < N$. . .

$\Gamma, X < Q, \Delta \vdash Q < N$. . .

$\Gamma, X < Q, \Delta \vdash X < N$. . .

To conclude (via the (Trans) rule):

$\Gamma, X < P, \Delta \vdash Q < N$ . . .

$\Gamma, X < P, \Delta \vdash P < Q$. (via weakening)

$\Gamma, X < P, \Delta \vdash P < N$. (via transitivity, applied to $Q$ itself)
Transitivity and Narrowing

- **Transitivity**: by inner induction on the derivation $\Gamma \vdash S <: Q$

  $\Gamma \vdash Q_1 <: S_1 \quad \Gamma, X <: Q_1 \vdash S_2 <: Q_2$

  $\Gamma \vdash \forall X <: S_1. S_2 <: \forall X <: Q_1. Q_2$

  $\Gamma \vdash T_1 <: Q_1 \quad \Gamma, X <: T_1 \vdash Q_2 <: T_2$

  $\Gamma \vdash \forall X <: Q_1. Q_2 <: \forall X <: T_1. T_2$

  To conclude (via the (All) rule):
Transitivity and Narrowing

- Transitivity: by inner induction on the derivation $\Gamma \vdash S <: Q$

\[
\begin{align*}
\Gamma \vdash Q_1 <: S_1 & \quad \Gamma, X <: Q_1 \vdash S_2 <: Q_2 \\
\Gamma, X <: Q_1 & \vdash S_2 <: \forall X <: Q_1.Q_2
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash T_1 <: Q_1 & \quad \Gamma, X <: T_1 \vdash Q_2 <: T_2 \\
\Gamma, X <: T_1 & \vdash Q_2 <: \forall X <: T_1.T_2
\end{align*}
\]

To conclude (via the (All) rule):

4 $\Gamma \vdash T_1 <: S_1$
Transitivity and Narrowing

- Transitivity: by inner induction on the derivation $\Gamma \vdash S <: Q$

\[
\begin{align*}
\Gamma \vdash Q_1 <: S_1 & \quad \Gamma, X <: Q_1 \vdash S_2 <: Q_2 \\
\Gamma \vdash \forall X <: S_1. S_2 <: \forall X <: Q_1. Q_2
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash T_1 <: Q_1 & \quad \Gamma, X <: T_1 \vdash Q_2 <: T_2 \\
\Gamma \vdash \forall X <: Q_1. Q_2 <: \forall X <: T_1. T_2
\end{align*}
\]

To conclude (via the (All) rule):

1. $\Gamma \vdash T_1 <: S_1$
2. $\Gamma, X <: T_1 \vdash S_2 <: T_2$ (via narrowing: $Q_1$ structurally smaller than $Q$)
Transitivity and Narrowing

- **Transitivity**: by inner induction on the derivation $\Gamma \vdash S <: Q$

\[
\begin{array}{c}
\Gamma \vdash Q_1 <: S_1 \\
\Gamma \vdash Q_2 <: S_2 \\
\Gamma, X <: Q_1 \vdash S_2 <: Q_2 \\
\Gamma \vdash \forall X <: S_1.S_2 <: \forall X <: Q_1.Q_2 \\
\end{array}
\]

To conclude (via the (All) rule):

1. $\Gamma \vdash T_1 <: S_1$
2. $\Gamma, X <: T_1 \vdash S_2 <: T_2$ (via narrowing: $Q_1$ structurally smaller than $Q$)

- **Narrowing**: by inner induction on $\Gamma, X <: Q, \Delta \vdash M <: N$

\[
\begin{array}{c}
\Gamma, X <: Q, \Delta \vdash Q <: N \\
\Gamma, X <: Q, \Delta \vdash X <: N \\
\end{array}
\]

To conclude (via the (Trans) rule):
Transitivity and Narrowing

- Transitivity: by inner induction on the derivation $\Gamma \vdash S <: Q$

  \[
  \begin{array}{c}
  \Gamma \vdash Q_1 <: S_1 \\
  \Gamma, X <: Q_1 \vdash S_2 <: Q_2 \\
  \hline
  \Gamma \vdash \forall X <: S_1.S_2 <: \forall X <: Q_1.Q_2
  \end{array}
  \quad
  \begin{array}{c}
  \Gamma \vdash T_1 <: Q_1 \\
  \Gamma, X <: T_1 \vdash Q_2 <: T_2 \\
  \hline
  \Gamma \vdash \forall X <: Q_1.Q_2 <: \forall X <: T_1.T_2
  \end{array}
  \]

  To conclude (via the (All) rule):
  \begin{enumerate}
  \item $\Gamma \vdash T_1 <: S_1$
  \item $\Gamma, X <: T_1 \vdash S_2 <: T_2$ (via narrowing: $Q_1$ structurally smaller than $Q$)
  \end{enumerate}

- Narrowing: by inner induction on $\Gamma, X <: Q, \Delta \vdash M <: N$

  \[
  \begin{array}{c}
  \Gamma, X <: Q, \Delta \vdash Q <: N \\
  \hline
  \Gamma, X <: Q, \Delta \vdash X <: N
  \end{array}
  \]

  To conclude (via the (Trans) rule):
  \begin{enumerate}
  \item $\Gamma, X <: P, \Delta \vdash Q <: N$
  \end{enumerate}
Transitivity and Narrowing

- **Transitivity**: by inner induction on the derivation $\Gamma \vdash S <: Q$

  \[
  \frac{\ldots \quad \Gamma \vdash Q_1 <: S_1 \quad \Gamma, X <: Q_1 \vdash S_2 <: Q_2}{\Gamma \vdash \forall X <: S_1. S_2 <: \forall X <: Q_1. Q_2}
  \]

  \[
  \frac{\ldots \quad \Gamma \vdash T_1 <: Q_1 \quad \Gamma, X <: T_1 \vdash Q_2 <: T_2}{\Gamma \vdash \forall X <: Q_1. Q_2 <: \forall X <: T_1. T_2}
  \]

  To conclude (via the (All) rule):
  1. $\Gamma \vdash T_1 <: S_1$
  2. $\Gamma, X <: T_1 \vdash S_2 <: T_2$ (via narrowing: $Q_1$ structurally smaller than $Q$)

- **Narrowing**: by inner induction on $\Gamma, X <: Q, \Delta \vdash M <: N$

  \[
  \frac{\ldots \quad \Gamma, X <: Q, \Delta \vdash Q <: N}{\Gamma, X <: Q, \Delta \vdash X <: N}
  \]

  To conclude (via the (Trans) rule):
  1. $\Gamma, X <: P, \Delta \vdash Q <: N$
  2. $\Gamma, X <: P, \Delta \vdash P <: Q$ (via weakening)

Alberto Ciaffaglione, Ivan Scagnetto
A weak HOAS approach to the POPLmark Challenge
Transitivity and Narrowing

Transitivity: by inner induction on the derivation $\Gamma \vdash S <: Q$

$\vdots$

$\Gamma \vdash Q_1 <: S_1$
$\Gamma, X <: Q_1 \vdash S_2 <: Q_2$

$\Gamma \vdash \forall X <: S_1.S_2 <: \forall X <: Q_1.Q_2$

$\vdots$

$\Gamma \vdash T_1 <: Q_1$
$\Gamma, X <: T_1 \vdash Q_2 <: T_2$

$\Gamma \vdash \forall X <: Q_1.Q_2 <: \forall X <: T_1.T_2$

To conclude (via the (All) rule):

1. $\Gamma \vdash T_1 <: S_1$
2. $\Gamma, X <: T_1 \vdash S_2 <: T_2$ (via narrowing: $Q_1$ structurally smaller than $Q$)

Narrowing: by inner induction on $\Gamma, X <: Q, \Delta \vdash M <: N$

$\vdots$

$\Gamma, X <: Q, \Delta \vdash Q <: N$

$\Gamma, X <: Q, \Delta \vdash X <: N$

To conclude (via the (Trans) rule):

1. $\Gamma, X <: P, \Delta \vdash Q <: N$
2. $\Gamma, X <: P, \Delta \vdash P <: Q$ (via weakening)
3. $\Gamma, X <: P, \Delta \vdash P <: N$ (via transitivity, applied to $Q$ itself)
A faithful alternative formulation
A faithful alternative formulation

- **Auxiliary judgments** (Closure, Well-formedness):

\[
\begin{align*}
\text{dom}(\Gamma) & \triangleq \{X_1, \ldots, X_n\} \\
\text{closed}(T, \Gamma) & \triangleq \forall Y. \ Y \in \text{fv}(T) \Rightarrow \exists U. \langle Y, U \rangle \in \Gamma
\end{align*}
\]

- **Subtyping rules**:

\[
\begin{align*}
\text{ok}(\Gamma) \land \text{closed}(S, \Gamma) \Rightarrow \text{sub}(\Gamma, S, \text{Top}) \quad (\text{top}) \\
\text{ok}(\Gamma) \quad X \notin \text{dom}(\Gamma) \land \text{closed}(T, \Gamma) \Rightarrow \text{ok}(\Gamma, \langle X, T \rangle) \quad (\text{ok-pair})
\end{align*}
\]
A faithful alternative formulation

- Auxiliary judgments (Closure, Well-formedness):
  
  \[ \text{dom}(\Gamma) \triangleq \{ X_1, \ldots, X_n \} \quad \text{closed}(T, \Gamma) \triangleq \forall Y. Y \in \text{fv}(T) \Rightarrow \exists U. \langle Y, U \rangle \in \Gamma \]

  \[
  \begin{align*}
  & \text{ok}(\emptyset) \quad \text{(ok-\emptyset)} \\
  & \frac{\text{ok}(\Gamma) \quad X \notin \text{dom}(\Gamma) \quad \text{closed}(T, \Gamma)}{\text{ok}(\Gamma, \langle X, T \rangle)} \quad \text{(ok-pair)}
  \end{align*}
  \]

- Subtyping (some rules):
  
  \[
  \begin{align*}
  & \frac{\text{ok}(\Gamma) \quad \text{closed}(S, \Gamma)}{\text{sub}(\Gamma, S, \text{Top})} \quad \text{(top)} \\
  & \frac{\text{ok}(\Gamma) \quad \langle X, U \rangle \in \Gamma}{\text{sub}(\Gamma, X, X)} \quad \text{(var)}
  \end{align*}
  \]

  \[
  \begin{align*}
  & \frac{\text{sub}(\Gamma, T_1, S_1) \quad \text{ok}(\Gamma, \langle X, T_1 \rangle) \Rightarrow \text{sub}((\Gamma, \langle X, T_1 \rangle), S_2, T_2)}{\text{sub}(\Gamma, \forall X : S_1 \cdot S_2, \forall X : T_1 \cdot T_2)} \quad \text{(all)}
  \end{align*}
  \]
A faithful alternative formulation

- **Auxiliary judgments** (Closure, Well-formedness):

  \[
  \text{dom}(\Gamma) \triangleq \{X_1, \ldots, X_n\} \quad \text{closed}(T, \Gamma) \triangleq \forall Y. \ Y \in \text{fv}(T) \Rightarrow \exists U. \ \langle Y, U \rangle \in \Gamma
  \]

  \[
  \frac{\text{ok}(\emptyset)}{(\text{ok} \cdot \emptyset)} \quad \frac{\text{ok}(\Gamma) \quad X \notin \text{dom}(\Gamma) \quad \text{closed}(T, \Gamma)}{\text{ok}(\Gamma, \langle X, T \rangle)} \quad (\text{ok} \cdot \text{pair})
  \]

- **Subtyping** (some rules):

  \[
  \frac{\text{ok}(\Gamma) \quad \text{closed}(S, \Gamma)}{\text{sub}(\Gamma, S, \text{Top})} \quad (\text{top}) \quad \frac{\text{ok}(\Gamma) \quad \langle X, U \rangle \in \Gamma}{\text{sub}(\Gamma, X, X)} \quad (\text{var})
  \]

  \[
  \frac{\text{sub}(\Gamma, T_1, S_1) \quad \text{ok}(\Gamma, \langle X, T_1 \rangle)}{\text{sub}(\Gamma, \forall X <: S_1.S_2, \forall X <: T_1.T_2)} \quad (\text{all})
  \]

- **Reflexivity**  \(\text{ok}(\Gamma) \wedge \text{closed}(S, \Gamma) \Rightarrow \text{sub}(\Gamma, S, S)\)

- **Transitivity**  \(\text{sub}(\Gamma, S, Q) \wedge \text{sub}(\Gamma, Q, T) \Rightarrow \text{sub}(\Gamma, S, T)\)

- **Narrowing**  \(\text{sub}((\Gamma, \langle X, Q \rangle, \Delta), M, N) \wedge \text{sub}(\Gamma, P, Q) \Rightarrow \text{sub}((\Gamma, \langle X, P \rangle, \Delta), M, N)\)
The encoding issue in a LF

Variables (\(\alpha\)-conversion, capture-avoiding substitution)

Traditional solutions (e.g., de Bruijn indices, first-order variables)

Higher-Order Abstract Syntax (HOAS) encapsulates the complexity, thus providing a high level of abstraction: representation by metavariables (functional constructors; functional application)

Incompatibility between HOAS and inductive types

No "full" HOAS: (\(T \rightarrow T\) \(\rightarrow T\)) violates the positivity constraint

Lack of higher-order recursion and induction principles

No inductive representation: (\(\text{Var} \rightarrow T\) \(\rightarrow T\)) generates parasite terms

Difficulty to reason about concepts delegated to the metalanguage

New logics (e.g., Nominal Logic, \(\text{FO} \lambda \Delta \nabla\))

A more conservative approach

Weak HOAS

The Theory of Contexts
The encoding issue in a LF

- Variables ($\alpha$-conversion, capture-avoiding substitution)
The encoding issue in a LF

- Variables ($\alpha$-conversion, capture-avoiding substitution)
  - traditional solutions (e.g. de Bruijn indices, first-order variables)
Variables (\(\alpha\)-conversion, capture-avoiding substitution)
- traditional solutions (e.g. de Bruijn indices, first-order variables)
- Higher-Order Abstract Syntax (HOAS) encapsulates the complexity, thus providing an high level of abstraction: representation by metavariables (functional constructors; functional application)
The encoding issue in a LF

- Variables (\(\alpha\)-conversion, capture-avoiding substitution)
  - traditional solutions (e.g. de Bruijn indices, first-order variables)
  - Higher-Order Abstract Syntax (HOAS) encapsulates the complexity, thus providing an high level of abstraction: representation by metavariables (functional constructors; functional application)

- Incompatibility between HOAS and inductive types
The encoding issue in a LF

- Variables (α-conversion, capture-avoiding substitution)
  - traditional solutions (e.g. de Bruijn indices, first-order variables)
  - Higher-Order Abstract Syntax (HOAS) encapsulates the complexity, thus providing an high level of abstraction: representation by metavariables (functional constructors; functional application)
- Incompatibility between HOAS and inductive types
  - no “full” HOAS: \((T \rightarrow T) \rightarrow T\) violates the positivity constraint
  - lack of higher-order recursion and induction principles
  - no inductive representation: \((\text{Var} \rightarrow T) \rightarrow T\) generates parasite terms
  - difficulty to reason about concepts delegated to the metalanguage
The encoding issue in a LF

- Variables (\(\alpha\)-conversion, capture-avoiding substitution)
  - traditional solutions (e.g. de Bruijn indices, first-order variables)
  - Higher-Order Abstract Syntax (HOAS) encapsulates the complexity, thus providing an high level of abstraction: representation by metavariables (functional constructors; functional application)

- Incompatibility between HOAS and inductive types
  - no “full” HOAS: \((T \rightarrow T) \rightarrow T\) violates the positivity constraint
  - lack of higher-order recursion and induction principles
  - no inductive representation: \((Var \rightarrow T) \rightarrow T\) generates parasite terms
  - difficulty to reason about concepts delegated to the metalanguage

- New logics (e.g. Nominal Logic, FO\(\lambda^\Delta^\nabla\))
The encoding issue in a LF

- Variables (\( \alpha \)-conversion, capture-avoiding substitution)
  - traditional solutions (e.g. de Bruijn indices, first-order variables)
  - Higher-Order Abstract Syntax (HOAS) encapsulates the complexity, thus providing an high level of abstraction: representation by metavariables (functional constructors; functional application)

- Incompatibility between HOAS and inductive types
  - no “full” HOAS: \((T \to T) \to T\) violates the positivity constraint
  - lack of higher-order recursion and induction principles
  - no inductive representation: \((\text{Var} \to T) \to T\) generates parasite terms
  - difficulty to reason about concepts delegated to the metalanguage

- New logics (e.g. Nominal Logic, \(FO\lambda^\Delta\nabla\))

- A more conservative approach
  - weak HOAS
  - the Theory of Contexts
Encoding: types and type environments

Variables as metavariables of a parametric, non-inductive type:
Parameter Var: Set.

Types as terms of an inductive type:

Coercion var: Var >-> Tp.

Example:
∀ X : Top. X is encoded (fa top (fun X:Var => X)).

Type environments as lists of pairs (explicit encoding)
Definition envTp: Set := (list (Var * Tp)).

Alberto Ciaffaglione, Ivan Scagnetto
A weak HOAS approach to the POPLmark Challenge
**Encoding: types and type environments**

- Variables as metavariables of a **parametric, non-inductive** type:
  
  Parameter Var: Set.

Variables as metavariables of a parametric, non-inductive type:

Parameter Var: Set.

Types as terms of an inductive type:

Inductive Tp: Set := top: Tp | var: Var -> Tp
| arr: Tp -> Tp -> Tp
| fa : Tp -> (Var -> Tp) -> Tp.

Coercion var: Var >-> Tp.

Example: \( \forall X <: Top. X \) is encoded \((fa \ top \ (fun \ X:Var \Rightarrow \ X))\)
Encoding: types and type environments

- **Variables** as metavariables of a **parametric, non-inductive** type:
  Parameter Var: Set.
- **Types** as terms of an **inductive** type:
  Inductive Tp: Set := top: Tp | var: Var -> Tp
  | arr: Tp -> Tp -> Tp
  | fa : Tp -> (Var -> Tp) -> Tp.
  Coercion var: Var >-> Tp.
  **Example:** \(\forall X <: Top. X\) is encoded \((fa \ top \ (fun \ X:Var => \ X))\)
- **Type environments** as lists of pairs (explicit encoding)
  Definition envTp: Set := (list (Var * Tp)).
Encoding: subtyping
Encoding: subtyping

The “(non) occurrence” concept (isin stands for $X \in \text{fv}(T)$):

Inductive isin (X:Var): Tp -> Prop :=
  isin_var: isin X X
| isin_arr: forall S T:Tp,
  isin X S \lor isin X T -> isin X (arr S T)
| isin_fa : forall S:Tp, forall U:Var->Tp,
  isin X S \lor (forall Y:Var, ~X=Y -> isin X (U Y)) ->
  isin X (fa S U).
Encoding: subtyping

- The “(non) occurrence” concept (\textit{isin} stands for $X \in \text{fv}(T)$):

  
  Inductive \textit{isin} (X:Var): Tp \to Prop :=
  
  \hspace{1em} \text{isin\_var}: \text{isin} \ X \ \ X
  
  \hspace{1em} | \text{isin\_arr}: \forall S \ T:Tp, 
  
  \hspace{2em} \text{isin} \ X \ S \ \ \text{\slash} \ \text{isin} \ X \ T \ \to \ \text{isin} \ X \ (\text{arr} \ S \ T)
  
  \hspace{1em} | \text{isin\_fa}: \forall S:Tp, \forall U:Var\toTp, 
  
  \hspace{2em} \text{isin} \ X \ S \ \text{\slash} \ (\forall Y:Var, \ ~X=Y \ \to \ \text{isin} \ X \ (U \ Y)) \ \to
  
  \hspace{2em} \text{isin} \ X \ (\text{fa} \ S \ U).

- The auxiliary judgments: $X \notin \text{dom}(\Gamma)$ (\textit{Gfresh}), $\langle X, T \rangle \in \Gamma$ (\textit{isinG}), $\text{closed}(T, \Gamma)$ (\textit{Gclosed}), $\text{ok}(\Gamma)$ (\textit{okEnv})
## Encoding: subtyping

- The “(non) occurrence” concept (*isin* stands for $X \in \text{fv}(T)$):

  Inductive *isin* $(X:\text{Var}) : \text{Tp} \to \text{Prop} := \neg$
  
  \[
  \begin{align*}
  \text{isin}_\text{var} & : \text{isin} \ X \ X \\
  | \text{isin}_\text{arr} & : \forall S \ T : \text{Tp}, \\
  & \text{isin} \ X \ S \ \lor \ \text{isin} \ X \ T \ \to \ \text{isin} \ X \ (\text{arr} \ S \ T) \\
  | \text{isin}_\text{fa} & : \forall S : \text{Tp}, \forall U : \text{Var} \to \text{Tp}, \\
  & \text{isin} \ X \ S \ \lor \ (\forall Y : \text{Var}, \sim X \ = \ Y \ \to \ \text{isin} \ X \ (U \ Y)) \ \to \\
  & \text{isin} \ X \ (\text{fa} \ S \ U).
  \end{align*}
  \]

- The auxiliary judgments: $X \notin \text{dom}(\Gamma)$ (*Gfresh*), $\langle X, \ T \rangle \in \Gamma$ (*isinG*), $\text{closed}(T, \Gamma)$ (*Gclosed*), $\text{ok}(\gamma)$ (*okEnv*)

- Subtyping (*subTp*):

  Inductive *subTp* : $\text{envTp} \to \text{Tp} \to \text{Tp} \to \text{Prop} := \ldots$

  \[
  \begin{align*}
  | \text{sub}_\text{fa} & : \forall G : \text{envTp}, \forall S_1 T_1 : \text{Tp}, \forall S_2 T_2 : \text{Var} \to \text{Tp}, \\
  & \text{subTp} \ G \ T_1 \ S_1 \ \to \\
  & (\forall X : \text{Var}, \ \text{okEnv} \ (\text{cons} \ (X,T_1) \ G) \ \to \\
  & \text{subTp} \ (\text{cons} \ (X,T_1) \ G) \ (S_2 \ X) \ (T_2 \ X)) \ \to \\
  & \text{subTp} \ G \ (\text{fa} \ S_1 \ S_2) \ (\text{fa} \ T_1 \ T_2).
  \end{align*}
  \]
The Theory of Contexts

Introduction
System F
Formalization
Conclusion

Decidability of equality over variables
For any variables \( x \) and \( y \), it is always possible to decide whether \( x = y \) or \( x \neq y \):

Axiom LEM_Var: \( \forall X \ Y: \text{Var}, X = Y \lor \neg X = Y. \)

Freshness/Unsaturation
For any term \( M \), there exists a variable \( x \) which does not occur free in it:

Axiom unsat: \( \forall T: \text{Tp}, \exists X: \text{Var}, \neg \text{in} \ X \ T. \)

Extensionality
Two contexts are equal if they are equal on a fresh variable; i.e., if \( M(\cdot) = N(\cdot) \) and \( x \not\in \text{ho} M(\cdot) \), \( N(\cdot) \), then \( M(\cdot) = N(\cdot) \):

Axiom tp_ext: \( \forall X: \text{Var}, \forall S \ T: \text{Var} \mapsto \text{Tp}, (\text{notin}_\text{ho} X S) \rightarrow (\text{notin}_\text{ho} X T) \rightarrow (S X) = (T X) \rightarrow S = T. \)

\( \beta \)-expansion
It is always possible to split a term into a context applied to a variable; i.e., given a term \( M \) and a variable \( x \), there exists a context \( N(\cdot) \) such that \( N(x) = M \) and \( x \not\in N(\cdot) \):

Axiom tp_exp: \( \forall S: \text{Tp}, \forall X: \text{Var}, \exists S': \text{Var} \mapsto \text{Tp}, (\text{notin}_\text{ho} X S') \land S = (S' X). \)
The Theory of Contexts

1. **Decidability of equality over variables** For any variables $x$ and $y$, it is always possible to decide whether $x = y$ or $x \neq y$:

   Axiom LEM_Var: $\forall X \ Y : \text{Var}, X=Y \lor \neg X=Y$.
The Theory of Contexts

1. **Decidability of equality over variables** For any variables $x$ and $y$, it is always possible to decide whether $x=y$ or $x \neq y$:
   
   Axiom LEM_Var: $\forall X \ Y: \text{Var}, \ X=Y \lor \neg X=Y$.

2. **Freshness/Unsaturation** For any term $M$, there exists a variable $x$ which does not occur free in it:
   
   Axiom unsat: $\forall T: \text{Tp}, \ \exists X: \text{Var}, \ \neg \text{in} X \ T$. 

3. **Extensionality** Two contexts are equal if they are equal on a fresh variable; i.e., if $M(x) = N(x)$ and $x \not\in M(\cdot)$, $N(\cdot)$, then $M(\cdot) = N(\cdot)$:
   
   Axiom tp_ext: $\forall X: \text{Var}, \forall S \ T: \text{Var->Tp}, \ \neg \text{in}_h X \ S \rightarrow \neg \text{in}_h X \ T \rightarrow S = T$.

4. **β-expansion** It is always possible to split a term into a context applied to a variable; i.e., given a term $M$ and a variable $x$, there exists a context $N(\cdot)$ such that $N(x) = M$ and $x \not\in N(\cdot)$:

   Axiom tp_exp: $\forall S: \text{Tp}, \forall X: \text{Var}, \ \exists S': \text{Var->Tp}, \ \neg \text{in}_h X \ S' \land S = S'(X)$.
The Theory of Contexts

1. **Decidability of equality over variables** For any variables $x$ and $y$, it is always possible to decide whether $x = y$ or $x \neq y$:

   Axiom LEM_Var: $\forall X \ Y: \text{Var}, X = Y \lor \neg X = Y$.

2. **Freshness/Unsaturation** For any term $M$, there exists a variable $x$ which does not occur free in it:

   Axiom unsat: $\forall T: \text{Tp}, \exists X: \text{Var}, \neg \text{in} X T$.

3. **Extensionality** Two contexts are equal if they are equal on a fresh variable; i.e., if $M(x) = N(x)$ and $x \notin M(\cdot), N(\cdot)$, then $M(\cdot) = N(\cdot)$:

   Axiom tp_ext: $\forall X: \text{Var}, \forall S \ T: \text{Var} \rightarrow \text{Tp}$,

   \[ (\neg \text{in}_\text{ho} X \ S) \rightarrow (\neg \text{in}_\text{ho} X \ T) \rightarrow (S X) = (T X) \rightarrow S = T \]
The Theory of Contexts

1. **Decidability of equality over variables** For any variables $x$ and $y$, it is always possible to decide whether $x=y$ or $x \neq y$:

   Axiom LEM_Var: for all $X, Y : \text{Var}$, $X=Y \lor \lnot X=Y$.

2. **Freshness/Unsaturation** For any term $M$, there exists a variable $x$ which does not occur free in it:

   Axiom unsat: for all $T : \text{Tp}$, there exists $X : \text{Var}$, not in $X T$.

3. **Extensionality** Two contexts are equal if they are equal on a fresh variable; i.e., if $M(x) = N(x)$ and $x \notin M(\cdot) \land N(\cdot)$, then $M(\cdot) = N(\cdot)$:

   Axiom tp_ext: for all $X : \text{Var}$, for all $S, T : \text{Var} \rightarrow \text{Tp}$, $(\not\text{in}_\text{ho} X S) \rightarrow (\not\text{in}_\text{ho} X T) \rightarrow (S X) = (T X) \rightarrow S = T$.

4. **$\beta$-expansion** It is always possible to split a term into a context applied to a variable; i.e., given a term $M$ and a variable $x$, there exists a context $N(\cdot)$ such that $N(x) = M$ and $x \notin N(\cdot)$:

   Axiom tp_exp: for all $S : \text{Tp}$, for all $X : \text{Var}$, there exists $S' : \text{Var} \rightarrow \text{Tp}$, $(\not\text{in}_\text{ho} X S') \land S = (S' X)$.
The Theory of Contexts at work

⇒ Reasoning by **structural induction** over contexts
The Theory of Contexts at work

⇒ Reasoning by **structural induction over contexts**

**Example:** Monotonicity of "occurrence" $\in (\text{isin})$:

$$
x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)
$$
The Theory of Contexts at work

⇒ Reasoning by structural induction over contexts

Example: Monotonicity of “occurrence” \( \in (\textsf{isin}) \):

\[
x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)
\]

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:
The Theory of Contexts at work

⇒ Reasoning by structural induction over contexts

Example: Monotonicity of “occurrence” \( \in (\text{isin}) \):

\[
x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)
\]

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:

- a predicate \( \text{measure}(T(z)) \), which counts the number \( n \) of constructors occurring in \( T(z) \) (where \( z \not\in T(\cdot) \))
The Theory of Contexts at work

⇒ Reasoning by structural induction over contexts

Example: Monotonicity of “occurrence” ∈ (isin):

\[ x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot) \]

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:

- a predicate measure\( (T(z)) \), which counts the number \( n \) of constructors occurring in \( T(z) \) (where \( z \notin T(\cdot) \))
- complete induction over the natural number \( n \)
The Theory of Contexts at work

⇒ Reasoning by structural induction over contexts

Example: Monotonicity of “occurrence” $\in (\text{isin})$:

$$ x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot) $$

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:

- a predicate $\text{measure}(T(z))$, which counts the number $n$ of constructors occurring in $T(z)$ (where $z \notin T(\cdot)$)
- complete induction over the natural number $n$
- $\beta$-expansion, extensionality
The Theory of Contexts at work

⇒ Reasoning by **structural induction** over contexts

**Example:** Monotonicity of “occurrence” $\in (\text{isin})$:

$$x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)$$

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:

- a predicate $\text{measure}(T(z))$, which counts the number $n$ of constructors occurring in $T(z)$ (where $z \notin T(\cdot)$)
- complete induction over the natural number $n$
- $\beta$-expansion, extensionality

**Lemma (preliminary):**

$$z \notin T(\cdot) \land \text{measure}(T(z))=n \land x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)$$
The Theory of Contexts at work

⇒ Reasoning by **structural induction** over contexts

**Example:** Monotonicity of “occurrence” \( \in (\text{isin}) \):

\[ x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot) \]

We recover the capability of “mimicking” the application of a **higher-order induction principle** by means of:

- a predicate \( \text{measure}(T(z)) \), which counts the number \( n \) of constructors occurring in \( T(z) \) (where \( z \notin T(\cdot) \))
- complete induction over the natural number \( n \)
- \( \beta \)-expansion, extensionality

**Lemma (preliminary):**

\[ z \notin T(\cdot) \land \text{measure}(T(z)) = n \land x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot) \]

**Proof (complete induction on \( n \), inversion of \( \text{measure}(T(z)) = n \)):**
The Theory of Contexts at work

⇒ Reasoning by structural induction over contexts
Example: Monotonicity of “occurrence” $\in$ (isin):

$x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)$

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:

- a predicate measure($T(z)$), which counts the number $n$ of constructors occurring in $T(z)$ (where $z \notin T(\cdot)$)
- complete induction over the natural number $n$
- $\beta$-expansion, extensionality

Lemma (preliminary):

$z \notin T(\cdot) \land \text{measure}(T(z))=n \land x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)$

Proof (complete induction on $n$, inversion of measure($T(z))=n$):

1. $\beta$-expansion: $\exists T'(\cdot).\ T'(z)=T(z) \land z \notin T'(\cdot)$
The Theory of Contexts at work

⇒ Reasoning by *structural induction* over contexts

**Example:** Monotonicity of “occurrence” \( \in (\text{isin}) \):

\[
x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)
\]

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:

- a predicate \( \text{measure}(T(z)) \), which counts the number \( n \) of constructors occurring in \( T(z) \) (where \( z \notin T(\cdot) \))
- complete induction over the natural number \( n \)
- \( \beta \)-expansion, extensionality

**Lemma (preliminary):**

\[
z \notin T(\cdot) \land \text{measure}(T(z))=n \land x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot)
\]

**Proof (complete induction on \( n \), inversion of \( \text{measure}(T(z))=n \)):**

1. \( \beta \)-expansion: \( \exists T'(\cdot). \quad T'(z)=T(z) \land z \notin T'(\cdot) \)
2. extensionality: \( T'(\cdot)=T(\cdot) \)
The Theory of Contexts at work

⇒ Reasoning by structural induction over contexts

Example: Monotonicity of “occurrence” ∈ (isin):

\[ x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot) \]

We recover the capability of “mimicking” the application of a higher-order induction principle by means of:

- a predicate \( \text{measure}(T(z)) \), which counts the number \( n \) of constructors occurring in \( T(z) \) (where \( z \notin T(\cdot) \))
- complete induction over the natural number \( n \)
- \( \beta \)-expansion, extensionality

Lemma (preliminary):

\[ z \notin T(\cdot) \land \text{measure}(T(z)) = n \land x \in T(y) \land x \neq y \Rightarrow x \in T(\cdot) \]

Proof (complete induction on \( n \), inversion of \( \text{measure}(T(z)) = n \)):

1. \( \beta \)-expansion: \( \exists T'(\cdot).\ T'(z) = T(z) \land z \notin T'(\cdot) \)
2. extensionality: \( T'(\cdot) = T(\cdot) \)

We can “lift” structural information about \( T(\cdot) \) to the level of terms.
Let us consider the case, where \( \text{measure}(T(z)) = 1 \). . . then, inverting such hypothesis, we get (among other subcases) the hypothesis where \( T(z) = \text{Top} \). . . then, we apply \( \beta \)-expansion to \( \text{Top} \), yielding a context \( T'(\cdot) \equiv \lambda x : Tp.\text{Top} \). . . in particular, we have \( T(z) = \text{Top} = T'(z) \), whence we can infer \( T(z) = (\lambda x : Tp.\text{Top})z \). Finally, by extensionality we infer \( T(\cdot) = \lambda x : Tp.\text{Top} \).
The Theory of Contexts at work: an example

Let us consider the case, where $\text{measure}(T(z)) = 1$. 

Then, inverting such hypothesis, we get (among other subcases) the hypothesis where $T(z) = \text{Top}$. Then, we apply $\beta$-expansion to $\text{Top}$, yielding a context $T'(\cdot) \equiv \lambda x: Tp. \text{Top}$. In particular, we have $T(z) = \text{Top} = T'(z)$, whence we can infer $T(z) = (\lambda x: Tp. \text{Top})z$. Finally, by extensionality we infer $T'(\cdot) = \lambda x: Tp. \text{Top}$.

⇒ We have lifted structural information from first-order term to its higher-order counterpart.
The Theory of Contexts at work: an example

Let us consider the case, where \( \text{measure}(T(z)) = 1 \).

\[ \ldots \] then, inverting such hypothesis, we get (among other subcases) the hypothesis where \( T(z) = \text{Top} \).

Finally, by extensionality we infer \( T(z) = (\lambda x:Tp.\text{Top})z \).

\[ \Rightarrow \]

We have lifted structural information from first-order term to its higher-order counterpart.
The Theory of Contexts at work: an example

Let us consider the case, where $\text{measure}(T(z)) = 1$.

- . . . then, inverting such hypothesis, we get (among other subcases) the hypothesis where $T(z) = \text{Top}$.
- . . . then, we apply $\beta$-expansion to $\text{Top}$, yielding a context $T'(\cdot) \equiv \lambda x : Tp.\text{Top}$.

⇒ We have lifted structural information from first-order term to its higher-order counterpart.
The Theory of Contexts at work: an example

Let us consider the case, where $\text{measure}(T(z)) = 1$.

- ... then, inverting such hypothesis, we get (among other subcases) the hypothesis where $T(z) = \text{Top}$.
- ... then, we apply $\beta$-expansion to $\text{Top}$, yielding a context $T'(\cdot) \equiv \lambda x : Tp. \text{Top}$.
- ... in particular, we have $T(z) = \text{Top} = T'(z)$, whence we can infer $T(z) = (\lambda x : Tp. \text{Top})z$. 

Finally, by extensionality we infer $T(\cdot) = \lambda x : Tp. \text{Top}$.

⇒ We have lifted structural information from first-order term to its higher-order counterpart.
Let us consider the case, where $\text{measure}(T(z)) = 1$.

- ... then, inverting such hypothesis, we get (among other subcases) the hypothesis where $T(z) = \text{Top}$.
- ... then, we apply $\beta$-expansion to $\text{Top}$, yielding a context $T'(\cdot) \equiv \lambda x:Tp.\text{Top}$.
- ... in particular, we have $T(z) = \text{Top} = T'(z)$, whence we can infer $T(z) = (\lambda x:Tp.\text{Top})z$.
- Finally, by extensionality we infer $T(\cdot) = \lambda x:Tp.\text{Top}$. 

$\Rightarrow$ We have lifted structural information from first-order term to its higher-order counterpart.
Let us consider the case, where \( \text{measure}(T(z)) = 1 \).

- ... then, inverting such hypothesis, we get (among other subcases) the hypothesis where \( T(z) = \text{Top} \).
- ... then, we apply \( \beta \)-expansion to \( \text{Top} \), yielding a context \( T'(\cdot) \equiv \lambda x : Tp.\text{Top} \).
- ... in particular, we have \( T(z) = \text{Top} = T'(z) \), whence we can infer \( T(z) = (\lambda x : Tp.\text{Top})z \).
- Finally, by extensionality we infer \( T(\cdot) = \lambda x : Tp.\text{Top} \).

\( \Rightarrow \) We have lifted structural information from first-order term to its higher-order counterpart.
Formal development of the POPLmark Challenge

Alberto Ciaffaglione, Ivan Scagnetto

A weak HOAS approach to the POPLmark Challenge
Auxiliary properties:

Lemma Gclosed_lemma: \( \forall G:envTp, \forall S T:Tp, \text{subTp} G S T \rightarrow G\text{closed} S G \land G\text{closed} T G. \)

Lemma unsatG: \( \forall G:envTp, \exists X:Var, G\text{fresh} X G. \)

Fixpoint domGtoT (G:envTp):= match G with 
| nil => top 
| (X,T)::G' => (arr X (domGtoT G')) end.

Main properties (i.e. the POPLmark Challenge):

Lemma reflexivity: \( \forall T:Tp, \forall G:envTp, \text{okEnv} G \rightarrow G\text{closed} T G \rightarrow \text{subTp} G T T. \)

Theorem trans_narrow: \( \forall Q:Tp, (\forall S:Tp, \forall G:envTp, (\text{subTp} G S Q) \rightarrow \forall T:Tp, (\text{subTp} G Q T) \rightarrow (\text{subTp} G S T)) /\right \) 
\( (\forall G':envTp, \forall M N:Tp, (\text{subTp} G' M N) \rightarrow \forall D G:envTp, \forall X:Var, \forall P:Tp, G'=\text{app} D (\text{cons} (X,Q) G) \rightarrow \text{subTp} G P Q \rightarrow \text{subTp} (\text{app} D (\text{cons} (X,P) G)) M N). \)
Auxiliary properties:

- Lemma Gclosed_lemma: for all $G: envTp$, for all $S, T:Tp$,
  $\text{subTp } G \ S \ T \rightarrow G\text{closed } S \ G \land G\text{closed } T \ G$. 
Formal development of the POPLmark Challenge

Auxiliary properties:

- **Lemma Gclosed_lemma:** for all $G:envTp$, for all $S T:Tp$, $\text{subTp } G S T \rightarrow \text{Gclosed } S G \land \text{Gclosed } T G$.
- **Lemma unsatG:** for all $G:envTp$, exists $X:Var$, $\text{Gfresh } X G$.

Fixpoint $\text{domGtoT}(G:envTp)$:

- $\text{domGtoT}(\text{nil}) = \text{top}$
- $\text{domGtoT}((X,T)::G') = \text{arr X (domGtoT G')}$

Main properties (i.e. the POPLmark Challenge):

- **Lemma reflexivity:** for all $T:Tp$, for all $G:envTp$, $\text{okEnv } G \rightarrow \text{Gclosed } T G \rightarrow \text{subTp } G T T$.
- **Theorem trans_narrow:** for all $Q:Tp$, $(\forall S:Tp, \forall G:envTp, (\text{subTp } G S Q) \rightarrow (\forall T:Tp, (\text{subTp } G Q T) \rightarrow (\text{subTp } G S T))$ \land $(\forall G':envTp, \forall M N:Tp, (\text{subTp } G' M N) \rightarrow (\forall D G:envTp, \forall X:Var, \forall P:Tp, G'=\text{app } D (\text{cons } (X,Q) G) \rightarrow (\text{subTp } G P Q \rightarrow (\text{subTp } (\text{app } D (\text{cons } (X,P) G)) M N))$. 

Alberto Ciaffaglione, Ivan Scagnetto

A weak HOAS approach to the POPLmark Challenge
Auxiliary properties:

- Lemma Gclosed lemma: \( \forall G : \text{envTp}, \forall S, T : \text{Tp}, \) 
  \( \text{subTp} \ G \ S \ T \rightarrow \ G\text{closed} \ S \ G \land \ G\text{closed} \ T \ G. \)

- Lemma unsatG: \( \forall G : \text{envTp}, \exists X : \text{Var}, \ G\text{fresh} \ X \ G. \)

- Fixpoint domGtoT \( (G : \text{envTp}) := \) match \( G \) with 
  | nil \( \rightarrow \) top 
  | (X, T)::G’ \( \rightarrow \) (arr X (domGtoT G’)) end.
Formal development of the POPLmark Challenge

Auxiliary properties:

- Lemma Gclosed_lemma: \( \forall G : \text{env} \text{Tp}, \forall S T : \text{Tp}, \exists G \text{closed}_G S G /\ G \text{closed}_G T G \).
- Lemma unsatG: \( \forall G : \text{env} \text{Tp}, \exists X : \text{Var}, \text{Gfresh}_X G \).
- Fixpoint domGtoT \( (G : \text{env} \text{Tp}) \):= match G with
  1. nil => top
  2. (X, T)::G' => (arr X (domGtoT G'))

Main properties (i.e. the POPLmark Challenge):

Alberto Ciaffaglione, Ivan Scagnetto
A weak HOAS approach to the POPLmark Challenge
Formal development of the POPLmark Challenge

Auxiliary properties:

- Lemma unsatG: forall G:envTp, exists X:Var, Gfresh X G.
- Fixpoint domGtoT (G:envTp):= match G with
  | nil => top
  | (X,T)::G' => (arr X (domGtoT G')) end.

Main properties (i.e. the POPLmark Challenge):

Auxiliary properties:

- Lemma Gclosed_lemma: for all G:envTp, for all S T:Tp, subTp G S T \rightarrow Gclosed S G \land Gclosed T G.
- Lemma unsatG: for all G:envTp, exists X:Var, Gfresh X G.
- Fixpoint domGtoT (G:envTp):= match G with
  \begin{align*}
  \mid \text{nil} & \Rightarrow \text{top} \\
  \mid (X,T)::G' & \Rightarrow (\text{arr} X (\text{domGtoT} G')) \text{ end.}
  \end{align*}

Main properties (i.e. the POPLmark Challenge):

- Lemma reflexivity: for all T:Tp, for all G:envTp, okEnv G \rightarrow Gclosed T G \rightarrow subTp G T T.
- Theorem trans_narrow: for all Q:Tp,
  \begin{align*}
  (\forall S:Tp, \forall G:envTp, \\
  (\text{subTp} G S Q) \rightarrow \forall T:Tp, (\text{subTp} G Q T) \rightarrow (\text{subTp} G S T)) \land \\
  (\forall G':envTp, \forall M N:Tp, (\text{subTp} G' M N) \rightarrow \\
  \forall D G:envTp, \forall X:\text{Var}, \forall P:Tp, \\
  G'=(\text{app} D (\text{cons} (X,Q) G)) \rightarrow \text{subTp} G P Q \rightarrow \\
  \text{subTp} (\text{app} D (\text{cons} (X,P) G)) M N).
  \end{align*}
POPLmark Challenge metrics of success
POPLmark Challenge metrics of success

- **Correctness.** An alternative presentation of System $F_{<:}$'s, equivalent to the original one but closer to the final formalization in Coq. The “on paper” translation to its formal counterpart is a matter of syntactic sugar, except for the use of weak-HOAS.
POPLmark Challenge metrics of success

- **Correctness.** An alternative presentation of System $F_{\text{\angle\angle}}$’s, equivalent to the original one but closer to the final formalization in Coq. The “on paper” translation to its formal counterpart is a matter of syntactic sugar, except for the use of weak-HOAS.

- **Reasonable overhead.** The weak HOAS approach frees the user from the burden of dealing with $\alpha$-conversion and capture-avoiding substitution of variables. The Theory of Contexts grants the extra ability to handle and reason about contexts (i.e. higher-order terms).
POPLmark Challenge metrics of success

- **Correctness.** An alternative presentation of System $F_{<:}$'s, equivalent to the original one but closer to the final formalization in Coq. The “on paper” translation to its formal counterpart is a matter of syntactic sugar, except for the use of weak-HOAS.

- **Reasonable overhead.** The weak HOAS approach frees the user from the burden of dealing with $\alpha$-conversion and capture-avoiding substitution of variables for variables. The Theory of Contexts grants the extra ability to handle and reason about contexts (i.e. higher-order terms).

- **Transparent technology.** The formal representation of System $F_{<:}$ and fundamental theorems are easily readable. The axioms of the Theory of Contexts are reminiscent of properties commonly taken for granted, by working with “pencil and paper”.

Alberto Ciaffaglione, Ivan Scagnetto

A weak HOAS approach to the POPLmark Challenge
POPLmark Challenge metrics of success

- **Correctness.** An alternative presentation of System F\(<\:)’s, equivalent to the original one but closer to the final formalization in Coq. The “on paper” translation to its formal counterpart is a matter of syntactic sugar, except for the use of weak-HOAS.

- **Reasonable overhead.** The weak HOAS approach frees the user from the burden of dealing with \(\alpha\)-conversion and capture-avoiding substitution of variables for variables. The Theory of Contexts grants the extra ability to handle and reason about contexts (i.e. higher-order terms).

- **Transparent technology.** The formal representation of System F\(<\:) and fundamental theorems are easily readable. The axioms of the Theory of Contexts are reminiscent of properties commonly taken for granted, by working with “pencil and paper”.

- **Reasonable cost of entry.** Coq is one of the best environments for a beginner in theorem proving: everybody is allowed to use fruitfully the proof assistant after a reasonable training effort.

Alberto Ciaffaglione, Ivan Scagnetto

A weak HOAS approach to the POPLmark Challenge
Related and future work
Several solutions collected at Challenge’s web page, carried out within the systems Abella, Alpha Prolog, ATS, Coq, Isabelle/HOL, Matita, Twelf via different approaches, which are classified as de Bruijn, Hybrid, HOAS, Locally nameless, Named variables, Nominal.
Related and future work

- Several solutions collected at Challenge’s web page, carried out within the systems Abella, Alpha Prolog, ATS, Coq, Isabelle/HOL, Matita, Twelf via different approaches, which are classified as de Bruijn, Hybrid, HOAS, Locally nameless, Named variables, Nominal

- The approach closest to ours is by the CMU group (in Twelf):

  \[
  \begin{align*}
  \text{tp: type.} & \quad \ldots \\
  \forall\text{forall: tp -> (tp -> tp) -> tp.}
  \end{align*}
  \]
Several solutions collected at Challenge’s web page, carried out within the systems Abella, Alpha Prolog, ATS, Coq, Isabelle/HOL, Matita, Twelf via different approaches, which are classified as de Bruijn, Hybrid, HOAS, Locally nameless, Named variables, Nominal

The approach closest to ours is by the CMU group (in Twelf):

\[ tp: \text{type. } \ldots \]
\[ \text{forall: } tp \rightarrow (tp \rightarrow tp) \rightarrow tp. \]

We keep sticking to Coq even in presence of frameworks which have a better support for HOAS (e.g. Twelf) and nominal calculi (e.g. Nominal Isabelle): the main proof is rather compact and follows closely the trace of its “informal” counterpart, carried out on paper
Several solutions collected at Challenge’s web page, carried out within the systems Abella, Alpha Prolog, ATS, Coq, Isabelle/HOL, Matita, Twelf via different approaches, which are classified as de Bruijn, Hybrid, HOAS, Locally nameless, Named variables, Nominal

The approach closest to ours is by the CMU group (in Twelf):

\[
\begin{align*}
tp & : \text{type}. \\
\text{forall} & : tp \rightarrow (tp \rightarrow tp) \rightarrow tp.
\end{align*}
\]

We keep sticking to Coq even in presence of frameworks which have a better support for HOAS (e.g. Twelf) and nominal calculi (e.g. Nominal Isabelle): the main proof is rather compact and follows closely the trace of its “informal” counterpart, carried out on paper.

From a pragmatic viewpoint, two remarks:

- lack of “smart” support for nested inductions (inconsistent cases automatically generated by the application of the induction tactic)
- handling the type environment could be seen as an overhead: the bookkeeping technique should provide a more compact formalization