# Toward a Theory of Contexts of Assumptions in Logical Frameworks

Alberto Momigliano

Department of Computer Science
Università degli Studi di Milano, Italy

Joint work with Amy Felty and Brigitte Pientka

# Background and motivation

- Logical frameworks: (mechanized) meta-logics for *representing*, *reasoning* and *programming* (over) formal systems.
- Our focus: Specifying formal systems using higher-order abstract syntax (HOAS):
  - binders in the object language $\Longleftrightarrow$ binders in meta-language.
- The 3 tenets of HOAS:
  - $\alpha$-renaming for free
  - substitution as $\beta$ reduction
  - contexts are *implicitly* handled via hypothetical parametric judgments
- The first two items are well understood, the third one somewhat less

This talk: let's revisit reasoning with assumptions/open objects

# More background and motivation

- Hypothetical judgments are well known and non controversial as far as *representation* is concerned since the late 80's ($\lambda$Prolog, Elf, Isabelle Pure)
- Less consensus on the *reasoning* side, different approaches along the Type/Proof Theory divide (and among TT as well. . . )

| Type-theorists | Proof-theorists |
|---|---|
| Twelf, Beluga, Delfin. . . | Abella, Tac, Hybrid . . . |

- An analogy: In the beginning, Gentzen created natural deduction, but then he switched to the sequent calculus in order to sort out the meta-theory.

# An Homage to ProofCert

- We all want to relate one framework to another with the aim to transfer theorems and proofs.
- There is ongoing work on relating TT and PT logical frameworks, mainly Minneapolis-based:
  - LF to $\lambda$Prolog
  - Twelf to $\mathcal{M}_2$ (only *closed* terms, so Twelf 1.2)
- . . . but issue of transferring reasoning in presence of assumptions is still unaddressed, e.g.
  - What is the logical status of Twelf's *regular world assumption*?
  - How do you map Beluga's contextual objects to a logic such as $\mathcal{G}$ or Coq?
  - . . .

# The rest of the talk

- Motivating examples
- Notation for contexts - not just a matter of style.
    - Contexts as structured sequences
    - Generalized contexts
    - Context relations
- Some **very** preliminary remarks about:
    - A unifying view of generalized context and ctx relation via the lattice of context assumptions;
    - the design of *ORBI* (Open challenge problem Repository for systems supporting reasoning with BInders), an intermediate language for specifying benchmarks problems.

# A first example: the polymorphic lambda-calculus

Grammar: Types and Terms - does not enforce scope

| | | | | | |
|---|---|---|---|---|---|
| Types $T$ | $::=$ | $\alpha$ | Terms M | $::=$ | $x$ |
| | | $\mid$ arr $T_1$ $T_2$ | | | $\mid$ lam $x$:$T.M$ $\mid$ app $M$ $N$ |
| | | $\mid$ all $\alpha.T$ | | | $\mid$ tlam $\alpha.M$ $\mid$ tapp $M$ $T$ |

# A first example: the polymorphic lambda-calculus

Grammar: Types and Terms - does not enforce scope

$$\text{Types } T \;::=\; \alpha \;\mid\; \text{arr}\, T_1\ T_2 \;\mid\; \text{all}\ \alpha.T$$

$$\text{Terms M} \;::=\; x \;\mid\; \text{lam}\ x{:}T.M \;\mid\; \text{app}\ M\ N \;\mid\; \text{tlam}\ \alpha.M \;\mid\; \text{tapp}\ M\ T$$

Alternative : Well-formed terms Martin-Löf-style - enforces scope

$$\dfrac{\dfrac{}{x \text{ term}}\, tm_v \\ \vdots \\ M \text{ term}}{(\text{lam}\, x.\, M) \text{ term}}\; tm_l^{x, tm_v} \qquad \dfrac{\dfrac{}{\alpha \text{ tp}}\, tp_v \\ \vdots \\ M \text{ term}}{(\text{tlam}\, \alpha.\, M) \text{ term}}\; tm_{tl}^{\alpha, tp_v}$$

$$\dfrac{M_1 \text{ term} \qquad M_2 \text{ term}}{(\text{app}\ M_1\ M_2) \text{ term}}\; tm_a \qquad \dfrac{M \text{ term} \qquad A \text{ tp}}{(\text{tapp}\ M\ A) \text{ term}}\; tm_{ta}$$

# More examples in context-free representation

Another example: some rules for "algorithmic" equality (the copy clause)

$$\frac{}{x \text{ term}}\ x \qquad \frac{}{\text{aeq } x\ x}\ ae_v$$

$$\vdots$$

$$\frac{\text{aeq } M\ N}{\text{aeq } (\text{lam}\, x.\, M)\ (\text{lam}\, x.\, N)}\ ae_l^{x,ae_v} \qquad \frac{\text{aeq } M_1\ N_1 \quad \text{aeq } M_2\ N_2}{\text{aeq } (\text{app } M_1\ M_2)\ (\text{app } N_1\ N_2)}\ ae_a$$

+ Context-free representation scales from grammars to judgments
- 2-dimensional notation is ambiguous
- Can we tell open vs. closed object?
- What about structural properties of assumptions? Shouldn't they be explicit?

# Putting things into context

What is a context?

# Putting things into context

What is a context?

- Typical answer [From Gentzen on]: A sequence – OK, sometimes a (multi)set – of formulas $A_1,\ A_2,\ \ldots, A_n$.

# Putting things into context

What is a context?

- Typical answer [From Gentzen on]: A sequence – OK, sometimes a (multi)set – of formulas $A_1,\ A_2,\ \ldots, A_n$.

Examples of contexts occurring in the above examples:

| | | |
|---|---|---|
| Type Context $\Gamma$ | $::=$ | $\cdot \mid \Gamma, \alpha\ \mathsf{tp}$ |
| Term/Type Context $\Gamma$ | $::=$ | $\cdot \mid \Gamma, \alpha\ \mathsf{tp} \mid \Gamma, x\ \mathsf{term}$ |
| Eq. Context $\Gamma$ | $::=$ | $\cdot \mid \Gamma, x\ \mathsf{term}, \mathsf{aeq}\ x\ x$ |

We are introducing the variable $x$ *together* with the assumption aeq $x$ $x$

# Putting things into context

What is a context?

- Typical answer [From Gentzen on]: A sequence – OK, sometimes a (multi)set – of formulas $A_1,\ A_2,\ \ldots, A_n$.

Examples of contexts occurring in the above examples:

| | | |
|---|---|---|
| Type Context $\Gamma$ | $::=$ | $\cdot \mid \Gamma, \alpha$ tp |
| Term/Type Context $\Gamma$ | $::=$ | $\cdot \mid \Gamma, \alpha$ tp $\mid \Gamma, x$ term |
| Eq. Context $\Gamma$ | $::=$ | $\cdot \mid \Gamma, x$ term, aeq $x\ x$ |

We are introducing the variable $x$ *together* with the assumption aeq $x\ x$

**Issue:** The use of ',' is ambiguous.

**Our view:** Contexts are structured sequences - distinguish between "blocks" and ctx using ';' and ','

This was already adopted in Twelf 1.3 with the notion of *regular world*

# Contexts as structured sequences

- A context is a sequence of declarations $D$ where a declaration is a block of individual atomic assumptions separated by ';', which binds tighter than ','.

| | | | |
|---|---|---|---|
| Atom | $A$ | | |
| Block of declaration | $D$ | $::=$ | $A \mid D; A$ |
| Context | $\Gamma$ | $::=$ | $\cdot \mid \Gamma, D$ |
| Schema | $S$ | $::=$ | $D_s \mid D_s \,\mathbf{|}\, S$ |

# Contexts as structured sequences

- A context is a sequence of declarations $D$ where a declaration is a block of individual atomic assumptions separated by ';', which binds tighter than ','.

$$\begin{array}{llcl} \text{Atom} & A & & \\ \text{Block of declaration} & D & ::= & A \mid D; A \\ \text{Context} & \Gamma & ::= & \cdot \mid \Gamma, D \\ \text{Schema} & S & ::= & D_s \mid D_s \mathbin{\textbf{|}} S \end{array}$$

- A *schema* classify a context, where '|' describes alternatives

$$\begin{array}{lcl} S_{\alpha x} & ::= & \alpha\ \text{tp} \mid x\ \text{term} \\ S_{xaeq} & ::= & x\ \text{term}; \text{aeq}\ x\ x \end{array}$$

- There are some obvious typing rules relating context and schemas, not shown here.
- **Convention:** $\Phi_{\alpha x}$ describes a context with schema $S_{\alpha x}$.

# Polymorphic lambda-calculus - revisited (with explicit context)

Well-formed Terms

$$\frac{x \text{ term} \in \Phi_{\alpha x}}{\Phi_{\alpha x} \vdash x \text{ term}}\ tm_v$$

$$\frac{\Phi_{\alpha x}, x \text{ term} \vdash M \text{ term}}{\Phi_{\alpha x} \vdash \text{lam}\, x.\, M \text{ term}}\ tm_l \qquad \frac{\Phi_{\alpha x} \vdash M_1 \text{ term} \quad \Phi_{\alpha x} \vdash M_2 \text{ term}}{\Phi_{\alpha x} \vdash (\text{app } M_1\ M_2) \text{ term}}\ tm_a$$

$$\frac{\Phi_{\alpha x}, \alpha \text{ tp} \vdash M \text{ term}}{\Phi_{\alpha x} \vdash \text{tlam}\, \alpha.\, M \text{ term}}\ tm_{tl} \qquad \frac{\Phi_{\alpha x} \vdash M \text{ term} \quad \Phi_{\alpha x} \vdash A \text{ tp}}{\Phi_{\alpha x} \vdash (\text{tapp } M\ A) \text{ term}}\ tm_{ta}$$

# Structural rules

- More fine-grained view of structural rules, which can be applied inside a block or to a whole ctx;
- Slightly unusual presentation of rules based on

  operations on declarations:

  - Let $\mathrm{rm}_A : S \to S'$ be a total function taking $D \in S$ and returning $D' \in S'$ where $D'$ is $D$ with $A$ removed, if $A \in D$; otherwise $D' = D$.
  - Let $\mathrm{perm}_\pi : S \to S'$ be a total function which permutes the elements of $D \in S$ according to $\pi$ to obtain $D' \in S'$.
- Note that we also "remove" whole declarations ($\mathrm{rm}_D$).
- This approach will hopefully pay off soon enough...

# Structural properties of declarations

- Declaration Weakening:

$$\frac{\Gamma, \mathrm{rm}_A(D), \Gamma' \vdash J}{\Gamma, D, \Gamma' \vdash J}\ \textit{d-wk}$$

- Declaration Strengthening:

$$\frac{\Gamma, D, \Gamma' \vdash J}{\Gamma, \mathrm{rm}_A(D), \Gamma' \vdash J}\ \textit{d-str}(\dagger)$$

with the proviso (†) that $A$ is irrelevant to $J$ (read *subordination*)

- Declaration Exchange

$$\frac{\Gamma, D, \Gamma' \vdash J}{\Gamma, \mathrm{perm}_\pi(D), \Gamma' \vdash J}\ \textit{d-exc}$$

# Structural properties of contexts

We canonically extended those operations to act on contexts ($\mathrm{rm}^*_A$, $\mathrm{perm}^*_\pi$):

- Context weakening

$$\frac{\mathrm{rm}^*_A(\Gamma) \vdash J}{\Gamma \vdash J}\ \textit{c-wk}$$

- Context strengthening

$$\frac{\Gamma \vdash J}{\mathrm{rm}^*_A(\Gamma) \vdash J}\ \textit{c-str}(\dagger)$$

- Context exchange

$$\frac{\Gamma \vdash J}{\mathrm{perm}^*_\pi(\Gamma) \vdash J}\ \textit{c-exc}$$

# Examples - revisited

Let's look back at the rule for well formed type application

$$\frac{\Phi_{\alpha x} \vdash M \text{ term} \quad ?? \vdash A \text{ tp}}{\Phi_{\alpha x} \vdash (\text{tapp } M\ A) \text{ term}}\ tm_{ta}$$

where $\Phi_{\alpha x} := \cdot \mid \Phi_{\alpha x}, x \text{ term} \mid \Phi_{\alpha x}, \alpha \text{ tp}$

# Examples - revisited

Let's look back at the rule for well formed type application

$$\frac{\Phi_{\alpha x} \vdash M \text{ term} \quad ?? \vdash A \text{ tp}}{\Phi_{\alpha x} \vdash (\text{tapp } M\ A) \text{ term}}\ tm_{ta}$$

where $\Phi_{\alpha x} := \cdot \mid \Phi_{\alpha x}, x \text{ term} \mid \Phi_{\alpha x}, \alpha \text{ tp}$

In what context is $A$ a well-formed type?

# Examples - revisited

Let's look back at the rule for well formed type application

$$\frac{\Phi_{\alpha x} \vdash M \text{ term} \quad ?? \vdash A \text{ tp}}{\Phi_{\alpha x} \vdash (\text{tapp } M\ A) \text{ term}}\ tm_{ta}$$

where $\Phi_{\alpha x} := \cdot \mid \Phi_{\alpha x}, x \text{ term} \mid \Phi_{\alpha x}, \alpha \text{ tp}$

In what context is $A$ a well-formed type?

1. $A$ is a type in $\Phi_{\alpha x}$, i.e., via implicit weakening: we use one **joint** context

# Examples - revisited

Let's look back at the rule for well formed type application

$$\frac{\Phi_{\alpha x} \vdash M \text{ term} \quad ?? \vdash A \text{ tp}}{\Phi_{\alpha x} \vdash (\text{tapp } M\ A) \text{ term}}\ tm_{ta}$$

where $\Phi_{\alpha x} := \cdot \mid \Phi_{\alpha x}, x \text{ term} \mid \Phi_{\alpha x}, \alpha \text{ tp}$

In what context is $A$ a well-formed type?

1. $A$ is a type in $\Phi_{\alpha x}$, i.e., via implicit weakening: we use one **joint** context
2. $A$ is a type in $\text{rm}^*_{x \text{ term}}(\Phi_{\alpha x})$: smallest context necessary, explicit strengthening.

# Examples - revisited

Let's look back at the rule for well formed type application

$$\frac{\Phi_{\alpha x} \vdash M \text{ term} \quad ?? \vdash A \text{ tp}}{\Phi_{\alpha x} \vdash (\text{tapp } M\ A) \text{ term}}\ tm_{ta}$$

where $\Phi_{\alpha x} := \cdot \mid \Phi_{\alpha x}, x \text{ term} \mid \Phi_{\alpha x}, \alpha \text{ tp}$

In what context is $A$ a well-formed type?

1. $A$ is a type in $\Phi_{\alpha x}$, i.e., via implicit weakening: we use one **joint** context
2. $A$ is a type in $\text{rm}^*_{x\ \text{term}}(\Phi_{\alpha x})$: smallest context necessary, explicit strengthening.

Why do we care?

# Examples - revisited

Let's look back at the rule for well formed type application

$$\frac{\Phi_{\alpha x} \vdash M \text{ term} \quad ?? \vdash A \text{ tp}}{\Phi_{\alpha x} \vdash (\text{tapp } M\ A) \text{ term}}\ tm_{ta}$$

where $\Phi_{\alpha x} := \cdot \mid \Phi_{\alpha x}, x \text{ term} \mid \Phi_{\alpha x}, \alpha \text{ tp}$

In what context is $A$ a well-formed type?

1. $A$ is a type in $\Phi_{\alpha x}$, i.e., via implicit weakening: we use one **joint** context
2. $A$ is a type in $\text{rm}^*_{x \text{ term}}(\Phi_{\alpha x})$: smallest context necessary, explicit strengthening.

Why do we care?
It's the meta-theory, stupid!

# Reasoning in contexts

**Attempt (Admissibility of Reflexivity)**

*For every term $M$, ??? $\vdash$ aeq $M$ $M$.*

The proof should be by induction on $M$...

Two possible approaches to fill that ???

1. Generalized context approach (G). The context used in the theorem contains all assumptions in the relevant judgments.
   Think Twelf/Beluga

# Reasoning in contexts

**Attempt (Admissibility of Reflexivity)**

*For every term M*, $??? \vdash \mathsf{aeq}\ M\ M$.

The proof should be by induction on $M$...

Two possible approaches to fill that ???

1. Generalized context approach (G). The context used in the theorem contains all assumptions in the relevant judgments.
   Think Twelf/Beluga
2. Context relations approach (R). State how the different relevant contexts are *related* (using $\mathsf{rm}^*$) and then state the theorem under the condition that the relation holds.
   Think Abella/Hybrid

# Generalized context : Reflexivity proof

- Here the *generalized* context has schema $x$ term; aeq $x$ $x$, so it's just $\Phi_{xaeq}$, but in general contains all the relevant assumptions from the contributing contexts.

**Theorem**

If $\Phi_{xaeq} \vdash M$ term then $\Phi_{xaeq} \vdash$ aeq $M$ $M$.

# Generalized context : Reflexivity proof

- Here the *generalized* context has schema $x$ term; aeq $x$ $x$, so it's just $\Phi_{xaeq}$, but in general contains all the relevant assumptions from the contributing contexts.

## Theorem

If $\Phi_{xaeq} \vdash M$ term then $\Phi_{xaeq} \vdash$ aeq $M$ $M$.

Proof.

```
rec ref : {ϕ:xaeqC}{M:[ϕ. term]} [ϕ. aeq (M ...) (M ...)] =
mlam ϕ ⇒ mlam M ⇒ case [ϕ. M ...] of
| [ϕ. #p.1 ...] ⇒ [ϕ. #p.2 ...]                          % Variable
| [ϕ. lam λx. M ... x] ⇒                                  % Lambda
  let  [ϕ,b:block y:term, ae_v:aeq y y. D ... b.1 b.2]=
         ref [ϕ, b:block y:term, ae_v:aeq y y] [ϕ, b. M ... b.1]
  in    [ϕ. ae_l λx. λw. (D ... x w)]
| [ϕ. app (M1 ...) (M2 ...)] ⇒                            % Application
  let  [ϕ. D1 ...] = ref [ϕ] [ϕ . M1 ... ] in
  let  [ϕ. D2 ...] = ref [ϕ] [ϕ . M2 ...] in [ϕ. ae_a (D1 ...) (D2 ...)];
```

□

# Context relations: Reflexivity

- Note that $\Phi_x = \mathrm{rm}^*_{\mathrm{aeq}\ x\ x}(\Phi_{xaeq})$. We can define the *graph* of this function inductively:

$$\frac{}{.\sim .}\ crel_e \qquad \frac{\Phi_x \sim \Phi_{xaeq}}{\Phi_x, x\ \mathrm{term} \sim \Phi_{xaeq}, x\ \mathrm{term}; \mathrm{aeq}\ x\ x}\ crel_{xaeq}$$

**Theorem**

Assume $\Phi_x \sim \Phi_{xaeq}$. If $\Phi_x \vdash M\ \mathrm{term}$ then $\Phi_{xaeq} \vdash \mathrm{aeq}\ M\ M$.

# Context relations: Reflexivity

- Note that $\Phi_x = \mathrm{rm}^*_{\mathrm{aeq}\ x\ x}(\Phi_{xaeq})$. We can define the *graph* of this function inductively:

$$\frac{}{.\sim .}\ crel_e \qquad \frac{\Phi_x \sim \Phi_{xaeq}}{\Phi_x, x\ \mathrm{term} \sim \Phi_{xaeq}, x\ \mathrm{term}; \mathrm{aeq}\ x\ x}\ crel_{xaeq}$$

## Theorem

Assume $\Phi_x \sim \Phi_{xaeq}$. If $\Phi_x \vdash M\ \mathrm{term}$ then $\Phi_{xaeq} \vdash \mathrm{aeq}\ M\ M$.

- In Abella (and Hybrid, give or take), the relations and the statement of the thm look like that:

```
Define xaeqR : olist → olist → prop by
          xaeqR  nil nil;
 nabla x, xaeqR (term x :: Ts) (aeq x x :: As) := xaeqR Ts As.

Theorem reflR: forall Ts As M, xaeqR Ts As →
                           {Ts |- term M} → {As |- aeq M M}.
```

# G-Promotion

We extend the previous example (algorithmic equality of terms) by considering *declarative* equality (which adds rules for reflexivity, symmetry, and transitivity), and prove them equivalent.

$$\begin{array}{rcl} \Phi_{xaeq} & := & \cdot \mid \Phi_{xaeq}, x \text{ term}; \text{aeq } x\ x \text{ (as seen before)} \\ \Phi_{xdeq} & := & \cdot \mid \Phi_{xdeq}, x \text{ term}; \text{deq } x\ x \\ \Phi_{xda} & := & \cdot \mid \Phi_{xda}, x \text{ term}; \text{deq } x\ x; \text{aeq } x\ x \end{array}$$

Recall the statement of reflexivity for terms:

*If* $\Phi_{xaeq} \vdash M \text{ term}$ *then* $\Phi_{xaeq} \vdash \text{aeq } M\ M$.

This lemma (and others) are needed in the proof of equivalence, but we must "promote" it first to the larger context $\Phi_{xda}$.

*If* $\Phi_{xda} \vdash M \text{ term}$ *then* $\Phi_{xda} \vdash \text{aeq } M\ M$.

# Proving Promotion

**Lemma**

*If* $\Phi_{xda} \vdash M$ term *then* $\Phi_{xda} \vdash$ aeq $M\ M$.

**Proof.**

| | |
|---|---|
| $\Phi_{xda} \vdash M$ term | by assumption |
| $\Phi_{xaeq} \vdash M$ term | by *c-str* |
| $\Phi_{xaeq} \vdash$ aeq $M\ M$ | by previous lemma |
| $\Phi_{xda} \vdash$ aeq $M\ M$ | by *c-wk* |

□

- In general, proofs of promotion for G versions of theorems require a combination of strengthening and weakening on contexts.
- R versions of promotion involve strengthening and weakening of one or both sides of a context relation.
- Wouldn't it be nice to have your logical framework support this?

# The semi-lattice of declarations

- Consider the set $\mathcal{D}$ of well-formed declarations and quotient it under the perm operation;
- Define $D_1 \preceq D_2$ iff there is $D$ s.t. $D_1 = \mathrm{rm}_D(D_2)$, modulo perm, which will ignore from now on.
- Define $D_1 \vee D_2$ as $\mathrm{remove\_dup}(D_1; D_2)$. Hence, "consing" (and cleaning up) two declarations yields their least upper bound.
- $\langle \mathcal{D}, \preceq, \epsilon, \bigvee \rangle$ is an upper semi-lattice with the empty declaration $\epsilon$ as zero element.
- Extend this construction to the set of schemas[1] and of well-formed ctx.

  *the generalized context in the G-version of thms can be seen as the* `lub` *of the relevant ctx*, *e.g.*

$$\Phi_{xda} \quad = \quad \Phi_{xaeq} \vee \Phi_{xdeq}$$

[1] Warning: we haven't worked out the details for alternatives yet.

# The lattice 2

- A picture: see white board

# The lattice 2

- A picture: see white board
- Phrasing weak/stren as "casting" – remember the promotion lemma:

$$\frac{\mathrm{rm}^*_D(\Gamma) \vdash J}{\Gamma \vdash J}\ \textit{c-wk} \quad \rightsquigarrow \quad \frac{\Gamma' \vdash J \qquad \Gamma' \preceq^c \Gamma}{\Gamma \vdash J}\ \textit{upc}$$

$$\frac{\Gamma \vdash J}{\mathrm{rm}^*_D(\Gamma) \vdash J}\ \textit{c-str}\dagger \quad \rightsquigarrow \quad \frac{\Gamma' \vdash J \qquad \Gamma \preceq^c \Gamma'}{\Gamma \vdash J}\ \textit{doc}$$

- What about ctx relations? The intuition is that we can recover the *certain* ctx relations by navigating the Hasse diagram.
- We conjecture that if you give us the G version of a thm involving a ctx $\Phi$, we can recover the R version by relating the ctxs of which $\Phi$ is the `lub`.

$$\text{Given } \Phi_{xda} = \Phi_{xaeq} \vee \Phi_{xdeq}, \text{ build } \Phi_{xaeq} \sim \Phi_{xdeq}$$

# ORBI

- We are designing Open challenge problem Repository for systems supporting reasoning with BInders, for sharing HOAS benchmark problems – Think an intermediate language between *OTT* and *TPTP*
- Uses a Beluga-like syntax enriched with *directives* so that the ORBI2X tools will compile it into legal Twelf/Beluga, Abella/Hybrid etc.

```
%Syntax
tm: type . app: tm → tm → tm. lam: (tm → tm) → tm.

%Judgments
aeq: tm → tm → type .

%Rules
ae_l: ({x:tm} aeq x x → aeq (M x) (N x)) → aeq (lam (λx. M x)) (lam (λx. N x))
     .

%Schemas
schema xaeqG: block (x:tm; u:aeq x x).
schema xaeqR: block (x:tm) ~ block (x:tm; u:aeq x x).

%Theorems
theorem reflG :  forall (Phi : xaeqG) (M : tm), [Phi |- aeq M M].
%PT explicit (M : tm) in reflG.
```

# Conclusions and future work

What started as a comparison work between HOAS systems is bearing additional fruits:

- A re-appraisal of the role of ctx in Proof Theory – in other terms a percolation of Beluga's type theory into PT, hopefully not scaring people away;
- The basis of a possible unification of how ctx are mechanized in TT and PT tools
- The design of an intermediate language for benchmark sharing

Current and future work:

- Carry out the G-to-R translation
- Work out the machinery to automate *promotion* lemmas

# The end

**Thank you!**

`http://complogic.cs.mcgill.ca/beluga/benchmarks/`