Type Soundness of λ-calculus with Shift/Reset and Let-Polymorphism

(Can we formalize "syntactic approach" in Isabelle/HOL + Nominal package?)

Noriko Hirota and Kenichi Asai

Ochanomizu University

September 4, 2009

History

Aug. 2005	Type soundness of monomorphic λ-calculus with shift/reset is formalized using Isabelle/HOL (without Nominal package). Tried to extend it to cope with let-polymorphism. But the α-renaming problem appeared to be too difficult.
Nov. 2006	Continued efforts without good progress.
Feb. 2007	I found Nominal package! I changed whole the proof accordingly, but the proof still did not complete.
late 2007	I found "Engineering Formal Metatheory" paper, and encouraged my student to use it to prove type soundness of our calculus.
Feb. 2008	The proof completed!
Feb. 2009	Resumed formalization with better Nominal package.
July 2009	Hit major(?) problem. (= this talk)

Special thanks to Christian Urban for numerous advice.

(Monomorphic) λ-calculus

Syntax $M = x \mid \lambda x.M \mid M@M$

Types $T = b \mid T \to T$

Typing rules

$$\Gamma, x : T \vdash x : T$$

$$\frac{\Gamma, x : T_1 \vdash M : T_2}{\Gamma \vdash \lambda x.M : T_1 \to T_2} \qquad \frac{\Gamma \vdash M_1 : T_1 \to T_2 \quad \Gamma \vdash M_2 : T_1}{\Gamma \vdash M_1 @ M_2 : T_2}$$

Soundness If $\vdash M : T$, then M is a value, or there exists M' such that M reduces to M' and $\vdash M' : T$.

Formalization If M is a closed program, we don't encounter α-renaming problem. Type soundness can be proved using "syntactic approach" without using nominal package.

(Monomorphic) λ-calculus with shift and reset

Syntax $M \;=\; x \mid \lambda x.M \mid M@M \mid \mathcal{S}k.M \mid \langle M \rangle$

Types $T \;=\; b \mid T_1/\alpha \rightarrow T_2/\beta$

Typing rules

$$\Gamma, x : T; \alpha \vdash x : T; \alpha$$

$$\frac{\Gamma, x : T_1; \alpha \vdash M : T_2; \beta}{\Gamma; \delta \vdash \lambda x.M : T_1/\alpha \rightarrow T_2/\beta; \delta} \qquad \frac{\Gamma, k : T/\delta \rightarrow \alpha/\delta; \sigma \vdash M : \sigma; \beta}{\Gamma; \alpha \vdash \mathcal{S}k.M : T; \beta}$$

$$\frac{\Gamma; \delta \vdash M_1 : T_1/\alpha \rightarrow T_2/\epsilon; \beta \quad \Gamma; \epsilon \vdash M_2 : T_1; \delta}{\Gamma; \alpha \vdash M_1 @ M_2 : T_2; \beta} \qquad \frac{\Gamma; \sigma \vdash M : \sigma; T}{\Gamma; \alpha \vdash \langle M \rangle : T; \alpha}$$

Soundness If $; \alpha \vdash M : T; \beta$, then M is a value or $\mathcal{S}k.M$ without surrounding reset, or there exists M' such that M reduces to M' and $; \alpha \vdash M' : T; \beta$.

Formalization We can still assume that M is a closed program, avoiding α-renaming problem. Type soundness can be proved using "syntactic approach" without using nominal package (3000 lines in Isabelle/HOL).

λ-calculus with let-polymorphism

Syntax $M \;=\; x \mid \lambda x.M \mid M@M \mid \text{let } x = M \text{ in } M$

Types $T \;=\; \alpha \mid b \mid T \to T$

Type scheme $S \;=\; T \mid \forall\alpha.S$

Typing rules $\Gamma, x : S \vdash x : T \quad S > T$

$$\frac{\Gamma, x : T_1 \vdash M : T_2}{\Gamma \vdash \lambda x.M : T_1 \to T_2} \qquad \frac{\Gamma \vdash M_1 : T_1 \to T_2 \quad \Gamma \vdash M_2 : T_1}{\Gamma \vdash M_1 @ M_2 : T_2}$$

$$\frac{\Gamma \vdash V : T_1 \quad \Gamma, x : \mathrm{close}(\Gamma, T_1) \vdash M : T_2}{\Gamma \vdash \text{let } x = V \text{ in } M : T_2}$$

(employing value restriction)

Soundness If $\vdash M : T$, then M is a value, or there exists M' such that M reduces to M' and $\vdash M' : T$.

Overview of required lemmas

- weakening lemma:
  If $\Gamma \vdash M : T$ and x free in M, then $\Gamma, x : S \vdash M : T$.
- instantiation lemma:
  If $\Gamma \vdash M : T$, then $\sigma(\Gamma) \vdash M : \sigma(T)$.
- substitution lemma:
  If $\Gamma, x : \forall\overline{\alpha}.T \vdash M : T'$ and $\Gamma \vdash V : T$ and $\overline{\alpha}$ is fresh in Γ, then $\Gamma \vdash M[x \mapsto V] : T'$.
- subject reduction (preservation):
  If $\Gamma \vdash M : T$ and M reduces to M', then $\Gamma \vdash M' : T$.
- progress

Weakening of let is subtle

- typing rule for let:

$$\frac{\Gamma \vdash V : T_1 \quad \Gamma, x : \mathrm{close}(\Gamma, T_1) \vdash M : T_2}{\Gamma \vdash \mathrm{let}\ x = V\ \mathrm{in}\ M : T_2}$$

- We have:

$$\frac{\vdash \lambda x.x : \alpha \to \alpha \quad f : \forall \alpha.\alpha \to \alpha \vdash f @ f : \beta \to \beta}{\vdash \mathrm{let}\ f = \lambda x.x\ \mathrm{in}\ f @ f : \beta \to \beta}$$

  but if we add $y : \alpha$ in the environment, $\mathrm{close}(y : \alpha, \alpha \to \alpha)$ becomes monomorphic $\alpha \to \alpha$, and the above expression no longer type checks.

- We actually need:

$$\frac{\Gamma \vdash V : T_1 \quad \Gamma, x : \mathrm{close}(\Gamma|_V, T_1) \vdash M : T_2}{\Gamma \vdash \mathrm{let}\ x = V\ \mathrm{in}\ M : T_2}$$

Substitution lemma gets difficult

For let case:

- From assumption, we have:

$$\frac{\Gamma' \vdash U : T_3 \quad \Gamma', y : \mathsf{close}(\Gamma'|_U, T_3) \vdash M : T_2}{\Gamma' \vdash \mathsf{let}\ y = U\ \mathsf{in}\ M : T_2}$$

  where $\Gamma' = \Gamma, x : \forall\overline{\alpha}.T_1$

- from induction hypothesis, we have:

$$\Gamma \vdash U[x \mapsto V] : T_3, \qquad \Gamma, y : \mathsf{close}(\Gamma'|_U, T_3) \vdash M[x \mapsto V] : T_2$$

  for $\Gamma \vdash V : T_1$.

- we have to show:

$$\frac{\Gamma \vdash U[x \mapsto V] : T_3 \quad \Gamma, y : \mathsf{close}(\Gamma|_{U[x \mapsto V]}, T_3) \vdash M[x \mapsto V] : T_2}{\Gamma \vdash (\mathsf{let}\ y = U\ \mathsf{in}\ M)[x \mapsto V] : T_2}$$

Can we prove:

from

$$\Gamma, y : \mathrm{close}(\Gamma'|_U, T_3) \vdash M[x \mapsto V] : T_2$$

the following

$$\Gamma, y : \mathrm{close}(\Gamma|_{U[x \mapsto V]}, T_3) \vdash M[x \mapsto V] : T_2$$

possibly using the lemma:

If $\Gamma, y : S \vdash M : T_2$ and $S' > S$, then $\Gamma, y : S' \vdash M : T_2$.

- But it seems the first $>$ below does not hold in general:

$$\mathrm{close}(\Gamma|_{U[x \mapsto V]}, T_3) \not> \mathrm{close}(\Gamma|_U, T_3) > \mathrm{close}(\Gamma'|_U, T_3)$$

- What if $\Gamma(z)$ contains type variables that have to be generalized in T_3, where z is a free variable of V?

Another α-renaming problem other than binders?

Typing rule for let:

$$\frac{\Gamma \vdash V : T_1 \quad \Gamma, x : \mathrm{close}(\Gamma, T_1) \vdash M : T_2}{\Gamma \vdash \mathrm{let}\ x = V\ \mathrm{in}\ M : T_2}$$

- Type variables that appear in $\Gamma \vdash V : T_1$ should be fresh.
- They can be substituted consistently.
- This is another instance of “variable convention.”
- But no binders are used here.

Another typing rule for let

- old:

$$\frac{\Gamma \vdash V : T_1 \quad \Gamma, x : \mathrm{close}(\Gamma, T_1) \vdash M : T_2}{\Gamma \vdash \mathrm{let}\ x = V\ \mathrm{in}\ M : T_2}$$

- new:

$$\frac{(\forall T_1 . S > T_1 \Rightarrow \Gamma \vdash V : T_1) \quad \Gamma, x : S \vdash M : T_2}{\Gamma \vdash \mathrm{let}\ x = V\ \mathrm{in}\ M : T_2}$$

Instantiation lemma gets difficult

If $\Gamma \vdash M : T$, then $\sigma(\Gamma) \vdash M : \sigma(T)$.

For let case:

- From assumption, we have:

$$\frac{(\forall T_1.S > T_1 \Rightarrow \Gamma \vdash V : T_1) \quad \Gamma, x : S \vdash M : T_2}{\Gamma \vdash \text{let } x = V \text{ in } M : T_2}$$

- from induction hypothesis, we have:

$$\forall T_1.S > T_1 \Rightarrow \sigma(\Gamma) \vdash V : \sigma(T_1) \qquad \sigma(\Gamma, x : S) \vdash M : \sigma(T_2)$$

- we have to show:

$$\frac{(\forall T_1'.\sigma(S) > T_1' \Rightarrow \sigma(\Gamma) \vdash V : T_1') \quad \sigma(\Gamma), x : \sigma(S) \vdash M : \sigma(T_2)}{\sigma(\Gamma) \vdash \text{let } x = V \text{ in } M : \sigma(T_2)}$$

Can we prove:

from

$$\forall T_1. S > T_1 \Rightarrow \sigma(\Gamma) \vdash V : \sigma(T_1)$$

the following

$$\forall T_1'. \sigma(S) > T_1' \Rightarrow \sigma(\Gamma) \vdash V : T_1' \quad ?$$

- Assume $\sigma(S) > T_1'$. I.e., pick any σ' such that $\sigma'(\sigma(S)) = T_1'$.
- If we can swap σ and σ', we have $\sigma(\sigma'(S)) = T_1'$ and hence $\sigma(S) > T_1' = \sigma(\sigma'(S))$.
- If σ is a bijection, we have $S > \sigma'(S)$.
- Then, from assumption (where $T_1 = \sigma'(S)$), we have $\sigma(\Gamma) \vdash V : \sigma(\sigma'(S))$ as desired.

Can we swap σ and σ'?

No.

- To prove the goal:

$$\forall T_1'.\sigma(S) > T_1' \Rightarrow \sigma(\Gamma) \vdash V : T_1'$$

  we have to consider all T_1', i.e., all σ'.
- We cannot assume that the chosen σ' is disjoint from σ.

To make σ' and σ disjoint, we need to weaken the typing rule for let:

$$\frac{(\forall T_1 \notin L.S > T_1 \Rightarrow \Gamma \vdash V : T_1) \quad \Gamma, x : S \vdash M : T_2}{\Gamma \vdash \text{let } x = V \text{ in } M : T_2}$$

Summary

"Engineering Formal Metatheory" approach in Coq:

- It went just fine.
- Treatment of mutual recursion was not clear.
- The choice of L did not go automatically.

Isabelle/HOL with Nominal package:

- Simple and fits well to intuition.
- In the α-renaming problem, there appears to be more than just binders.

Should I continue the proof?

- Formally proven at least in Coq.
- I do not have anyone expert in my building.
- (And writing proof scripts spoils my health.)