Adding Schematic Abstraction to λP

Ferruccio Guidi

HELM team, DISI, University of Bologna, Italy

ferruccio.guidi@unibo.it

February 28, 2018

1. Propositions as objects vs. propositions as types

- The encoding of logic into typed λ-calculus follows two paradigms: the so-called "propositions as objects" and "proposition as types".

Level	Propositions as objects		Propositions as types
Kind	$\star$		universe ($o \equiv \star$)
Type	universe (o)	assertion ($A\ \mathit{true}$)	proposition ($A \equiv A\ \mathit{true}$)
Object	proposition (A)	derivation (π)	derivation (π)

- Systems pursuing "propositions as objects": $\lambda{\rightarrow}$, AUT-68, LF, λP. Notice that *true* is a primitive function symbol of type: $o \rightarrow \star$.

- Systems pursuing "propositions as types": AUT-QE, System F, CC. Easier: we build propositions with the framework's type constructors.

- Nevertheless "propositions as types" requires stronger frameworks, *i.e.*, conjunction and disjunction have type $\star \rightarrow \star \rightarrow \star$ not in λP.

2. Predicative frameworks allowing propositions as types

- With "propositions as types" we need h.o. quantification of class $(\Box, \star)$ to represent logical rules with schematic propositional variables.

- $A, B \vdash A \wedge B$ becomes $\mathrm{land\_i} : (\forall A : \star)(\forall B : \star)(A \to B \to A \wedge B)$ and the quantification on B is of class $(\Box, \star)$ in λ-Cube terminology.

1. PTS-style impredicative solution: λC.

Add triples $(\Box, \Box, \Box)$ and $(\Box, \star, \star)$ to λP.

2. PTS-style predicative solution: henceforth λT (very powerful).

Add triples $(\Box, \Box, \Box)$ and $(\Box, \star, \Box)$ to λP.

3. PTS-style predicative solution: λQE $\approx$ AUT-QE (less powerful).

add triples $(\Box, \Box, \triangle)$, $(\Box, \star, \triangle)$, $(\Box, \triangle, \triangle)$, $(\star, \triangle, \triangle)$ to λP.

4. Refined PTS-style predicative solution: refined λQE $\approx$ AUT-QE.

Add parameter pairs $(\Box, \Box)$, $(\Box, \star)$, $(\star, \Box)$, $(\star, \star)$ to λP.

3. Discussion on the predicative frameworks

- System λT (solution 2) allows to write powerful constructions, *i.e.*, logical rules with schematic variables for connectives. Is this useful?

- The quantification $(\star, \square, \square)$ of λT can be seen both as internal and as schematic. Thus the former can precede the latter in constructions.

- Is it always the case that internal quantifications preceding schematic ones in constructions (rejected in λQE) can be thought as schematic?

- In λT and λQE instantiated assertions live in $\star$ while assertions with h.o. schematic variables live in $\square$ or $\triangle$, *i.e.*, at a different level.

- Reasonably, de Bruijn's unified binder $[x : \alpha]$ emerges as a device to accommodate schematic abstraction in Automath-related languages.

- The refined λ-Cube (solution 4) pursues the syntactic distinction between internal abstraction (Π, λ) and schematic abstraction (¶, §).

4. Introducing the system $\lambda\Upsilon\mathrm{P}$: a step towards $\lambda_\infty \oplus \lambda\mathrm{P}$

- Here we are proposing to develop a framework in which λ_∞ provides for the schematic abstraction while $\lambda\mathrm{P}$ provides for the internal one.

- In the perspective of the refined λ-Cube we are proposing mainly to unify ¶ and § in $(\Upsilon x : \alpha)$, inspired by $[x : \alpha]$ (differing from Π and λ).

- In the ideal $\lambda_\infty \oplus \lambda\mathrm{P}$ the two subsystems are independent (contrary to $\lambda\mathrm{QE}$), so schematic and internal abstractions can be mixed in terms.

- By meeting the requirement of independence, we conjecture that our system can have a simple structure and uniform validity rules like $\lambda\mathrm{T}$.

- The ideal $\lambda_\infty \oplus \lambda\mathrm{P}$ supports constructions like schematic variables for connectives without the hybrid quantification $(\star, \square, \square)$ of $\lambda\mathrm{T}$.

- To start with, we are proposing here the system $\lambda\Upsilon\mathrm{P}$ that extends $\lambda\mathrm{P}$ with the h.o. schematic abstraction provided by the Υ quantifier.

5. Syntax and conversion in $\lambda\Upsilon\mathrm{P}$

- Our system has the syntax of simplified LF with three levels of terms (kinds K, families T and objects M) and one category for contexts L.

$$H,\,K \ ::= \star \mid (\Pi n : U).K \mid (\Upsilon u : H).K$$

$$T,\,U \ ::= u \mid (\Pi n : U).T \mid (\lambda n : U).T \mid (N).T \mid (\Upsilon u : H).T \mid (U).T$$

$$M,\,N ::= n \mid (\lambda n : U).M \mid (N).M \mid (\Upsilon u : H).M \mid (U).M$$

$$L \quad ::= \circ \mid L.(n : U) \mid L.(u : H)$$

- We add a h.o abstraction $(\Upsilon u : H)$ for objects, families and kinds. We add the corresponding application (U) for objects and families.

- To the refined λ-Cube $(\Upsilon u : H).T$ is a ¶ and a § at the same time.

- Moreover we pose that $(U).(\Upsilon u : H)$ is a β-redex giving rise to:

$$L \vdash (U).(\Upsilon u : H).M =_\beta [U/u].M \qquad L \vdash (U).(\Upsilon u : H).T =_\beta [U/u].T$$

6. Validity in $\lambda\Upsilon$P

- The judgments (LF): $\vdash L\,!$ (L is valid), $L \vdash K\,!$ (K is valid in L), $L \vdash T : K$ (T belongs to K in L), $L \vdash M : T$ (M belongs to T in L).

- Here we omit the validity rules concerning the LF fragment of $\lambda\Upsilon$P.

$$\frac{L \vdash H\,! \quad L.(u:H) \vdash K\,!}{L \vdash (\Upsilon u:H).K\,!}\,1 \qquad \frac{L \vdash H\,! \quad L.(u:H) \vdash T : K}{L \vdash (\Upsilon u:H).T : (\Upsilon u:H).K}\,2 \qquad \frac{L \vdash H\,! \quad L.(u:H) \vdash M : T}{L \vdash (\Upsilon u:H).M : (\Upsilon u:H).T}\,3$$

$$\frac{L \vdash U : H \quad L \vdash T : (\Upsilon u:H).K}{L \vdash (U).T : [U/u].K}\,4 \qquad \frac{L \vdash U : H \quad L \vdash M : (\Upsilon u:H).T}{L \vdash (U).M : [U/u].T}\,5$$

$$\frac{L \vdash M : T_1 \quad L \vdash T_1 =_\beta T_2 \quad L \vdash T_2 : (\Upsilon u:H).K}{L \vdash M : T_2}\,6 \qquad \frac{L \vdash U : \star \quad L.(n:U) \vdash T : (\Upsilon u:H).K}{L \vdash (\Pi n:U).T : (\Upsilon u:H).K}\,7$$

- Rules 6 an 7 show that in a PTS for $\lambda\Upsilon$P there is a sort for each $(\Upsilon u:H).K$. Moreover $(\Upsilon u:H).T$ is a $(\Pi u:H).T$ with Π-reduction.

- In the perspective of λQE, we break the sort $\triangle$ in a system of sorts $\triangle_{H,K} : \Box$, that are as many as the simple types from one base type.

7. Validity in $\lambda\Upsilon$P continued

- Note: $L \vdash T : \triangle_{H,K}$ gives more information on T than $L \vdash T : \triangle$.

- The "start" rules come from LF hence $L \vdash n : T$ implies $L \vdash T : \star$, Therefore n cannot take $(\Upsilon u : H).M$, which is is not a first-class object.

- The ideal $\lambda_\infty \oplus \lambda$P must have a "start" rule to remove this limitation.

- Instead $L \vdash u : H$ implies $L \vdash H$! therefore u can take $(\Upsilon u : H).T$.

- Interesting properties to prove for $\lambda\Upsilon$P include strong normalization of valid terms. Confluence and safety of reduction should be PTS-like.

- Strong normalization should be reducible to the one of $\lambda\delta$-2, *i.e.*, $\lambda\!\rightarrow$-like, like strong normalization of λC is reducible to the one of $\lambda\omega$.

- The ideal $\lambda_\infty \oplus \lambda$P must also include the f.o. schematic abstraction $(\Upsilon n : U)$ with which we enable the quantification $(\star, \triangle, \triangle)$ of λQE.

- It is quite likely that we need to consider the kind $(\Upsilon n : U).K$ a sort.

8. Testing $\lambda\Upsilon$P on the "Grundlagen"

- Statement: any logical framework claiming to support "propositions as types" must accept a translation of the "Grundlagen der Analysis".
- Among the realistic fragments of math formalized with "propositions as types", the "Grundlagen" does not need very expressive frameworks.
- We took the λProlog version of the "Grundlagen" for CC. We turned f.o. quantification to $(\Pi n : U)$ and h.o. quantification to $(\Upsilon u : H)$.
- Notice that by so doing, $(\Pi n : U)$ precedes $(\Upsilon u : H)$ in some cases.
- We implemented an efficient validator for $\lambda\Upsilon$P in λProlog, which operates in the $\mathcal{L}_\lambda^\beta$ fragment and never unwinds its reduction machine.
- Typical runs of three validators on the ELPI engine (same hardware): lyp [$\lambda\Upsilon$P] (9.4s), Helena [$\approx \lambda\delta$-3] (35.7s), ALT-0/PTS [CC] (43.7s).
- The interactions of Υ and Π in $\lambda\Upsilon$P should clarify the design of $\lambda\delta$-3.

References

[1] C. Dunchev, F. Guidi, C. Sacerdoti Coen, and E. Tassi. ELPI: fast, Embeddable, λProlog Interpreter. In M. Davis, A. Fehnker, A. McIver, and A. Voronkov, editors, *Proceedings of 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-20)*, volume 9450 of *Lecture Notes in Computer Science*, pages 460–468, Berlin, Germany, December 2015. Springer.

[2] F. Guidi. Verified Representations of Landau's "Grundlagen" in the $\lambda\delta$ Family and in the Calculus of Constructions. *Journal of Formalized Reasoning*, 8(1):93–116, December 2015.

[3] F. Guidi. The Formal System $\lambda\Upsilon$P. Technical Report AMS Acta 5754, University of Bologna, Bologna, Italy, January 2018.

Thank you