Higher-Order Unification with Definition by Cases

Chad E. Brown[1], David M. Cerna[2]

[1]Czech Technical University in Prague
[2]Czech Academy of Science Institute for Computer Science

August 12th 2022

Outline

- We revisit extensions of the simply typed lambda calculus by restrictions of *Hilbert's choice operator*.

$$\forall p : \alpha \to o . \forall x : \alpha . (p\ x \Rightarrow p\ (\varepsilon^{\alpha} p))$$

- We introduce a restrict which differs from earlier investigations where both
  - an **algorithm** and
  - type (**unitary**)

  were provided.
- We show that our restriction differs from the earlier approach as it is at least type **finitary**.
- In addition, we show that the space of solutions is significantly different from unification over both the lambda calculus and earlier investigations.

Simply Typed λ-Calculus

- Simple Types
  - ι (individuals)
  - $\alpha \to \beta$ (function types)
- Simply typed λ-terms (no logic):
  - Typed Variables x
  - Typed Constants c (optional)
  - Applications $s\ t$
  - Abstractions $\lambda x.s$
- $\beta\eta$-equivalence (unification is undecidable)

Henkin Semantics

- Frame: nonempty set $\mathcal{D}_\alpha$ for each type α.
- $\mathcal{D}_{\alpha\to\beta} \subseteq (\mathcal{D}_\beta)^{\mathcal{D}_\alpha}$ for types α, β
- (A frame is "standard" if $=$ instead of $\subseteq$.)
- combinatorial closure
- Assignment φ: map variables x of type α to $\varphi x \in \mathcal{D}_\alpha$.
- Each term t of type α evaluates as $\mathcal{I}_\varphi\ t \in \mathcal{D}_\alpha$.
- $\mathcal{I}\ (s\ t)$ denotes function application of $\mathcal{I}\ s$ to $\mathcal{I}\ t$.
- A Henkin model is a frame and an $\mathcal{I}$ (determined by its interpretation of constants).
- Let $\mathcal{H}$ be a class of Henkin models.
- Two terms s and t of type α are semantically equivalent if $\mathcal{I}_\varphi s = \mathcal{I}_\varphi t$ for all Henkin models in $\mathcal{H}$ and assignments φ.

Higher-Order Logic

- Simply Typed λ-Calculus
- Plus base type o (propositions/booleans/truth values)
- Plus Logic (via logical constants and properties of those constants)
- Plus More (sometimes)

Higher-Order ATP

- Automated theorem proving in HOL is hard.
- Benchmark: TH0 part of TPTP Library
- Used for higher-order division of CASC.
- Many HO ATPs use Huet's preunification
  - Designed for simply typed λ-calculus without logic, not for higher-order logic.
- TH0 allows for a choice operator $\varepsilon_\alpha : (\alpha \to o) \to \alpha$
- Restricting to Henkin models interpreting the logic and choice operators results in
  - More semantically equivalent terms
  - unsolvable unification problems become solvable.
- With choice, one can define if-then-else.
- With if-then-else, one can define a "cases" operator

Motivating Example

- Consider the problem: $f\ (Xa)\ (Xb) = f\ b\ a$
- Not unifiable without choice.
- HO ATPs use some form of choice when solving
- LEO-III:

```
...
thf(12,plain,((((@+ [A:$i]: (((a = a) => (A = b)) & ((a = b) => (A = a)))) != b) | (((@+ [A:$i]: (((b =
a) => (A = b)) & ((b = b) => (A = a)))) != a)),introduced(choice instance)).
thf(14,plain,((((@+ [A:$i]: ((A = b) & ((a = b) => (A = a)))) != b) | (((@+ [A:$i]: (((b = a) => (A = b))
& (A = a))) != a)),inference(simp,[status(thm)],[12])).
thf(18,plain,(! [A:$i] : ((~ ((A = b) & ((a = b) => (A = a)))) | ((((@+ [B:$i]: ((B = b) & ((a = b) => (B
= a)))) = b) & ((a = b) => (((@+ [B:$i]: ((B = b) & ((a = b) => (B = a)))) = a))))),inference(choice,
[status(esa)],[17])).
...
```

- Zipperposition:

```
...
thf(zip_derived_cl11, plain, (![X0 : $o > $i, X1 : $o > $i, X2 : $i > $o]: (~ (X2 @ a) | (((^[Y0 : $o]:
((^[Y1 : $o,Y2 : $i]: (X0 @ Y1)) @ Y0 @ (X1 @ Y0))) @  ($true)) != (b))| (((^[Y0 : $i]: ((^[Y1 :
$o,Y2 : $i]: (X0 @ Y1)) @ (X2 @ Y0) @ Y0)) @  b) != (a)))), inference('fluid_loob_hoist',
[status(thm)], [zip_derived_cl5])).
thf(zip_derived_cl13, plain, (![X0 : $o > $i, X2 : $i > $o]: (~ (X2 @ a) | ((X0 @ ($true)) != (b)) |
((X0 @ (X2 @ b)) != (a)))), inference('ho_norm', [status(thm)], [zip_derived_cl11])).
thf(zip_derived_cl127, plain, (![X0 : $o > $i]:(((X0 @ ($true)) != (b)) | ((X0 @  ((^[Y0 : $i]:
((((^[Y1 : $i]: (Y1)) @ Y0) = ((^[Y1 : $i]: (a)) @ Y0)))) @ b))!= (a)))),
inference('fluid_log_symbol_hoist', [status(thm)], [zip_derived_cl13])).
...
```

Beeson's if-then-else

- In *"Unification in Lambda-Calculi with if-then-else"* Beeson introduces a **d** operator and reduction relation
  - $\mathbf{d}(x, x, a, b) = a$
  - $\mathbf{d}(x, y, Zy, Zx) = Zx$
  - $\mathbf{d}(x, y, y, x) = x$
  - $\mathbf{d}(x, y, a, a) = a$
- Beeson's **d** allows for less committing solutions.
- Consider the unification problem $Xa =^? a$
  - $\theta = \{X \mapsto \lambda x.x\}$ $\quad$ $\theta = \{X \mapsto \lambda x.a\}$
- Beeson's **d** allows $\theta = \{X \mapsto \lambda z.\mathbf{d}\ z\ a\ a\ Yz\}$
- This solution generalizes the previous two.
- Beeson introduced **d** with the goal of compression and developing a unitary theory.
- Our goal is analysis and development of a theory where more term pairs are unifiable.

Simply Typed λ-Calculus With Cases

- Idea: Drop logic and return to simply typed λ-calculus
- Add constants $\mathsf{d}_\alpha : \iota \to \iota \to \alpha \to \alpha \to \alpha$.
- Semantic restriction:
  - $\mathcal{I}_\varphi$ d a b u $w = u$ if $\mathcal{I}_\varphi a = \mathcal{I}_\varphi b$
  - $\mathcal{I}_\varphi$ d a b u $w = w$ if $\mathcal{I}_\varphi a \neq \mathcal{I}_\varphi b$
- Restrict to Henkin models intepretating d
  - Henkin models with d
- The unification problem is now between simply typed λ-calculus and higher-order logic with choice.
- **Question:** What is the nature of this unification problem?

Back to the Motivating example

- $f\ (Xa)\ (Xb) = f\ \ b\ a$ has four d solutions
  - $\theta_1 X = \lambda z.\mathrm{d}^{\iota}\ z\ a\ b\ a$.
  - $\theta_2 X = \lambda z.\mathrm{d}^{\iota}\ z\ b\ a\ b$.
  - $\theta_3 X = \lambda z.\mathrm{d}^{\iota}\ z\ a\ b\ (\mathrm{d}^{\iota}\ z\ b\ a\ (Y\ z))$.
  - $\theta_4 X = \lambda z.\mathrm{d}^{\iota}\ z\ b\ a\ (\mathrm{d}^{\iota}\ z\ a\ b\ (Y\ z))$.
- θ_3 and θ_4 are equi-general:

$$(\lambda z.\mathrm{d}^{\iota}\ z\ a\ b\ (\mathrm{d}^{\iota}\ z\ b\ a\ (Y\ z)))$$

$$=_{\iota\iota}$$

$$(\lambda z.\mathrm{d}^{\iota}\ z\ b\ a\ (\mathrm{d}^{\iota}\ z\ a\ b\ (Y\ z))).$$

- Has a single solution most general solution.
- follows Beeson's construction.
- This need not be the case.

Multiple Solutions

- It is unclear how our d operator can be used to compress the solutions of

$$(\lambda z.X\ z\ z) = (\lambda z.z)$$

- but there are more interesting **examples**!
- Consider

$$f\ (\lambda u.X\ u\ u)\ (\lambda u.X\ u\ a) = f\ (\lambda u.f\ (g\ u)\ a)\ (\lambda u.f\ (g\ u)\ u)$$

- The only solution to $\lambda u.X\ u\ a = \lambda u.f\ (g\ u)\ u$ is
  - $\theta\ X = \lambda z_1 z_2.f\ (g\ z_1)\ z_1$
- This does not solve $\lambda u.X\ u\ u = \lambda u.f\ (g\ u)\ a$
- d we get $\theta X = \lambda z_1 z_2.\mathrm{d}^{\iota}\ z_1\ z_2\ (f\ (g\ z_1)\ a)\ (f\ (g\ z_1)\ z_1)$.

Multiple Solutions

- The solution

$$\theta X = \lambda z_1 z_2.\mathrm{d}^{\iota}\ z_1\ z_2\ (f\ (g\ z_1)\ a)\ (f\ (g\ z_1)\ z_1).$$

  works because

  - $\forall u.u = a \to f\ (g\ u)\ a = f\ (g\ u)\ u$ is valid in all Henkin models, and thus

$$(\lambda u.\mathrm{d}^{\iota}\ u\ a\ (f\ (g\ u)\ a)\ (f\ (g\ u)\ u)) = (\lambda u.f\ (g\ u)\ u)$$

    is valid in all Henkin models of d.

- We can also check if the second argument is a:

$$\theta X = \lambda z_1 z_2.\mathrm{d}^{\iota}\ z_2\ a\ (f\ (g\ z_1)\ z_1)\ (f\ (g\ z_1)\ a).$$

- We can take this construction even further!

Complex Solutions

- Consider

$$f\ (\lambda u.X\ u\ u)\ (\lambda u.X\ u\ (h\ u))$$

$$=$$

$$f\ (\lambda u.f\ (g\ u)\ (h\ u))\ (\lambda u.f\ (g\ u)\ u)$$

- We are now forced to use both arguments:

$$\theta X = \lambda z_1 z_2.\mathrm{d}^{\iota}\ z_2\ (h\ z_1)\ (f\ (g\ z_1)\ z_1)\ (f\ (g\ z_1)\ (h\ z_2))$$

- $\forall u.u = h\ u \rightarrow f\ (g\ u)\ u = f\ (g\ u)\ (h\ u)$ is valid in all Henkin interpretations, entailing that

- $(\lambda u.\mathrm{d}^{\iota}\ u\ (h\ u)\ (f\ (g\ u)\ u)\ (f\ (g\ u)\ (h\ u))) =$ $(\lambda u.f\ (g\ u)\ (h\ u))$ is valid in all Henkin models of D.

Future Work

- We introduce a restriction of *Hilbert's choice operator*
- We consider unification over simply-typed lambda calculus.
- We show that our restriction differs from previous work (Beeson's d).
- We plan to investigate algorithmic approaches to the problem,
- and resolve our conjecture concerning the type of the of theory.