Programming and proving with classical types

Cristina Matache[†,‡]
Joint work with Victor Gomes[‡] and Dominic Mulligan[‡]

[†]University of Oxford
[‡]University of Cambridge

APLAS 2017

Motivation

- Proof assistants:
  - Logic: intuitionistic vs. classical;
  - Evidence: explicit (witness) vs. implicit.

Question

Classical proof assistant with explicit evidence.

Motivation

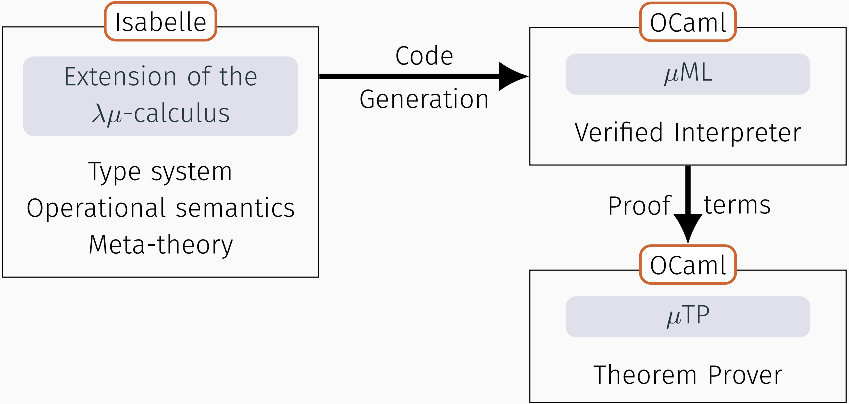
Question

Classical proof assistant with explicit evidence.

- Problems:
  - Logic: classical first-order;
  - Evidence: $\lambda\mu$ terms.

Outline

1. Evidence: $\lambda\mu$ and μML
2. Realisation: μTP



Evidence: $\lambda\mu$ and μML

$\rho, \sigma, \tau ::= \bot \mid \tau \to \sigma$	types
$t, r, s ::= x \mid \lambda x{:}\sigma.t \mid t\ s \mid$	λ-calculus terms
$\mu\alpha{:}\sigma.c$	μ-abstraction
$c ::= [\alpha]t$	named terms

bind continuation α

apply α to term t

$$\mu\alpha{:}\sigma.[\alpha]t$$

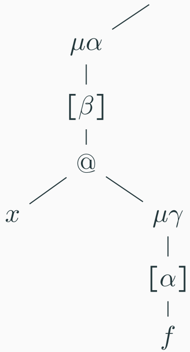
type the continuation expects

Reduction Example

$$(\mu\alpha.[\beta](x\ \mu\gamma.[\alpha]f))\ y \longrightarrow \mu\alpha.[\beta](x\ \mu\gamma.[\alpha](f\ y))$$

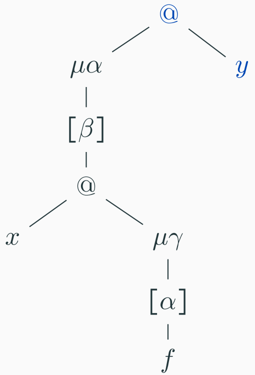
Reduction Example

$$(\mu\alpha.[\beta](x\ \mu\gamma.[\alpha]f))\ y \longrightarrow \mu\alpha.[\beta](x\ \mu\gamma.[\alpha](f\ y))$$



Reduction Example

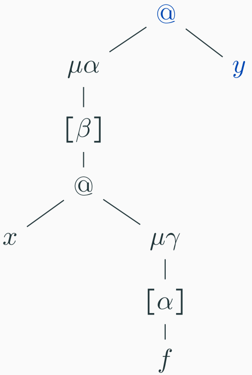
$$(\mu\alpha.[\beta](x\ \mu\gamma.[\alpha]f))\ y \longrightarrow \mu\alpha.[\beta](x\ \mu\gamma.[\alpha](f\ y))$$



Reduction Example

$$(\mu\alpha.[\beta](x\ \mu\gamma.[\alpha]f))\ y \longrightarrow \mu\alpha.[\beta](x\ \mu\gamma.[\alpha](f\ y))$$

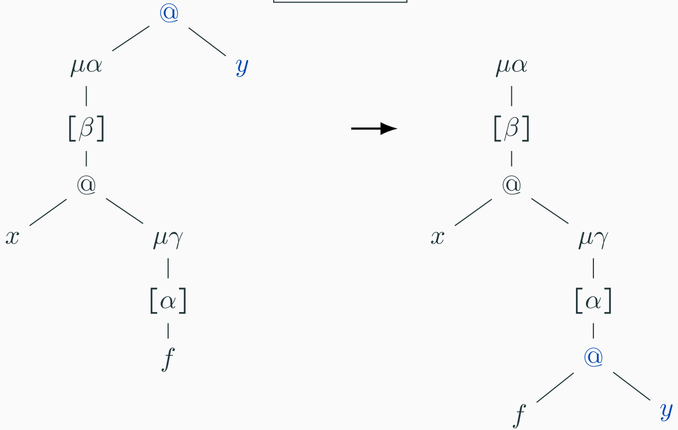
$$\boxed{\alpha \mapsto \square\ y}$$



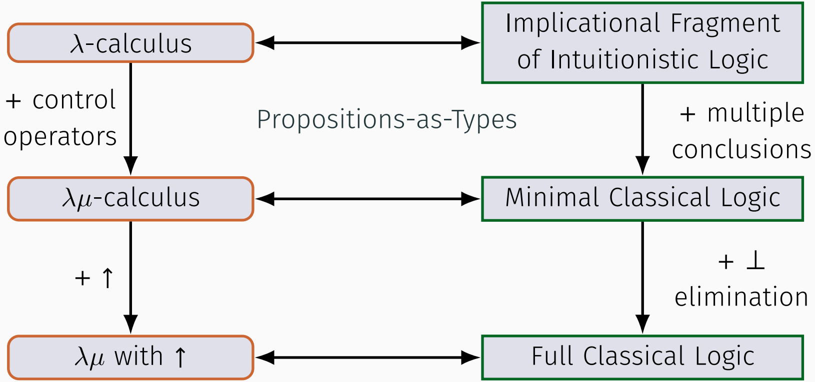
Reduction Example

$$(\mu\alpha.[\beta](x\ \mu\gamma.[\alpha]f))\ y \longrightarrow \mu\alpha.[\beta](x\ \mu\gamma.[\alpha](f\ y))$$

$$\boxed{\alpha \mapsto \square\ y}$$



The Propositions-as-Types Correspondence



Extending $\lambda\mu$

- Open terms for classical tautologies $\Longrightarrow$ Add $\uparrow$

  [Ariola & Herbelin, 2003]

$\boxed{\neg\neg A \to A}$ $\quad \neg A \equiv A \to \bot$

$$\Gamma; \Delta \vdash t : A \;\longleftrightarrow\; \Gamma \vdash A; \Delta$$

$$\frac{\Gamma; \Delta, \alpha{:}A \vdash_c c}{\Gamma; \Delta \vdash \mu\alpha{:}A.c : A}$$

$$\frac{\Gamma; \Delta \vdash t : A \qquad \alpha{:}A \in \Delta}{\Gamma; \Delta \vdash_c [\alpha]t}$$

$$\frac{\Gamma; \Delta \vdash t : \bot}{\Gamma; \Delta \vdash_c [\uparrow]t}$$

($\bot$ elimination)

Extending $\lambda\mu$

- Open terms for classical tautologies $\Longrightarrow$ Add $\uparrow$

  [Ariola & Herbelin, 2003]

$$\boxed{\neg\neg A \to A} \qquad \neg A \equiv A \to \bot$$

$$\Gamma; \Delta \vdash t : A \;\longleftrightarrow\; \Gamma \vdash A; \Delta$$

$$\lambda y{:}\neg\neg A$$

$$\frac{\Gamma; \Delta, \alpha{:}A \vdash_c c}{\Gamma; \Delta \vdash \mu\alpha{:}A.c : A}$$

$$\frac{\Gamma; \Delta \vdash t : A \qquad \alpha{:}A \in \Delta}{\Gamma; \Delta \vdash_c [\alpha]t}$$

$$\frac{\Gamma; \Delta \vdash t : \bot}{\Gamma; \Delta \vdash_c [\uparrow]t}$$

($\bot$ elimination)

Extending $\lambda\mu$

- Open terms for classical tautologies $\Longrightarrow$ Add $\uparrow$
  [Ariola & Herbelin, 2003]

$$\boxed{\neg\neg A \to A} \qquad \neg A \equiv A \to \bot$$

$$\Gamma; \Delta \vdash t : A \;\longleftrightarrow\; \Gamma \vdash A; \Delta$$

$$\lambda y{:}\neg\neg A$$
$$|$$
$$\mu\alpha{:}A$$

$$\frac{\Gamma; \Delta, \alpha{:}A \vdash_c c}{\Gamma; \Delta \vdash \mu\alpha{:}A.c : A}$$

$$\frac{\Gamma; \Delta \vdash t : A \qquad \alpha{:}A \in \Delta}{\Gamma; \Delta \vdash_c [\alpha]t}$$

$$\frac{\Gamma; \Delta \vdash t : \bot}{\Gamma; \Delta \vdash_c [\uparrow]t}$$

($\bot$ elimination)

Extending $\lambda\mu$

- Open terms for classical tautologies $\Longrightarrow$ Add $\uparrow$

  [Ariola & Herbelin, 2003]

$\boxed{\neg\neg A \to A}$ $\quad \neg A \equiv A \to \bot$

$$\Gamma; \Delta \vdash t : A \quad \longleftrightarrow \quad \Gamma \vdash A; \Delta$$

$$\frac{\Gamma; \Delta, \alpha{:}A \vdash_c c}{\Gamma; \Delta \vdash \mu\alpha{:}A.c : A}$$

$$\frac{\Gamma; \Delta \vdash t : A \qquad \alpha{:}A \in \Delta}{\Gamma; \Delta \vdash_c [\alpha]t}$$

$$\frac{\Gamma; \Delta \vdash t : \bot}{\Gamma; \Delta \vdash_c [\uparrow]t}$$

($\bot$ elimination)

$\lambda y{:}\neg\neg A$

|

$\mu\alpha{:}A$

|

$[\uparrow]$

Extending $\lambda\mu$

- Open terms for classical tautologies $\Longrightarrow$ Add $\uparrow$

  [Ariola & Herbelin, 2003]

$\boxed{\neg\neg A \to A}$ $\quad \neg A \equiv A \to \bot$

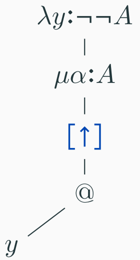
$$\Gamma;\Delta \vdash t : A \quad\longleftrightarrow\quad \Gamma \vdash A;\Delta$$

$$\frac{\Gamma;\Delta,\alpha{:}A \vdash_c c}{\Gamma;\Delta \vdash \mu\alpha{:}A.c : A}$$

$$\frac{\Gamma;\Delta \vdash t : A \qquad \alpha{:}A \in \Delta}{\Gamma;\Delta \vdash_c [\alpha]t}$$

$$\frac{\Gamma;\Delta \vdash t : \bot}{\Gamma;\Delta \vdash_c [\uparrow]t}$$

($\bot$ elimination)



Extending $\lambda\mu$

- Open terms for classical tautologies $\Longrightarrow$ Add $\uparrow$
  [Ariola & Herbelin, 2003]

$\boxed{\neg\neg A \to A}$ $\qquad \neg A \equiv A \to \bot$

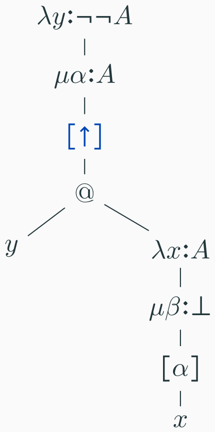
$$\Gamma; \Delta \vdash t : A \;\longleftrightarrow\; \Gamma \vdash A; \Delta$$

$$\frac{\Gamma; \Delta, \alpha{:}A \vdash_c c}{\Gamma; \Delta \vdash \mu\alpha{:}A.c : A}$$

$$\frac{\Gamma; \Delta \vdash t : A \qquad \alpha{:}A \in \Delta}{\Gamma; \Delta \vdash_c [\alpha]t}$$

$$\frac{\Gamma; \Delta \vdash t : \bot}{\Gamma; \Delta \vdash_c [\uparrow]t}$$

($\bot$ elimination)



Extending $\lambda\mu$

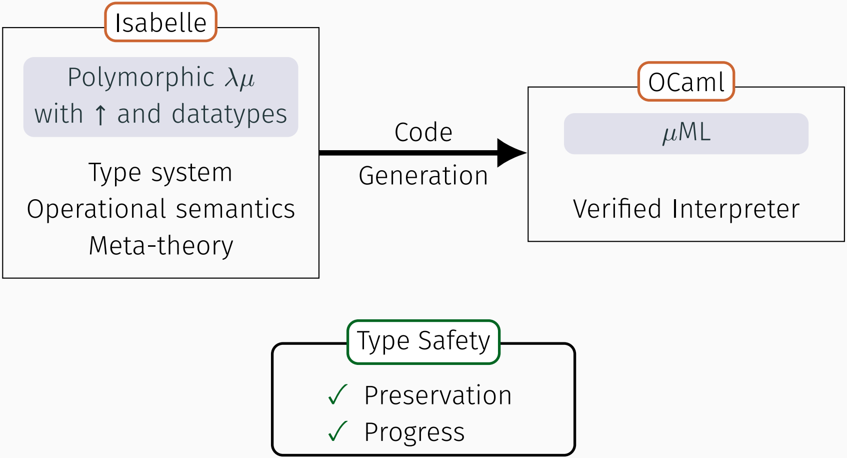
- First order quantification:

$$\rho, \sigma, \tau ::= \ \dots \mid a \mid \forall a.\sigma \qquad \text{types}$$

$$t, r, s ::= \ \dots \mid \Lambda a.t \qquad \text{terms}$$

- Datatype encoding not unique $\Longrightarrow$ Built-in datatypes
  - natural numbers and primitive recursion [Geuvers et. al., 2013]
  - booleans
  - products
  - tagged unions

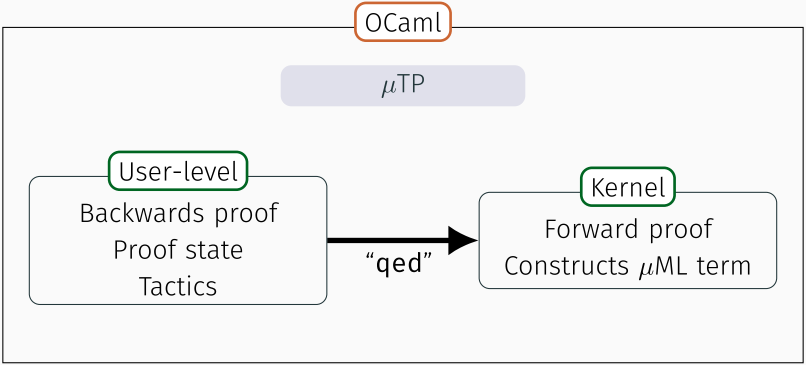
μML Interpreter



Realisation: μTP

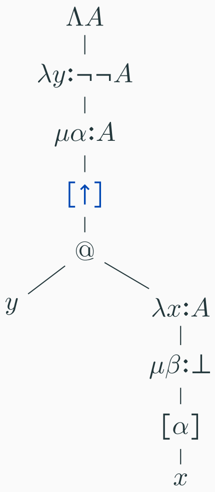
μTP Theorem Prover

- LCF-style theorem prover
- Use μML terms as evidence



Example μTP Proof

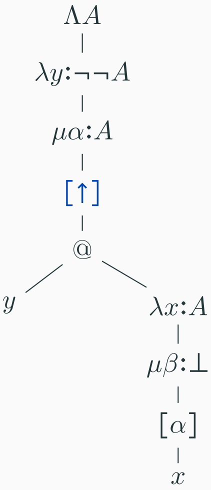
$$\boxed{\Lambda A.\ \neg\neg A \to A}$$




```
conjecture
  (mk_all_t
    (mk_arrow_t
      (mk_arrow_t (mk_arrow_t
        (mk_var_t 0) mk_bot_t) mk_bot_t)
      (mk_var_t 0)));
  apply 0 all_intro_tac;
  apply 0 imp_intro_tac;
  apply 0 mu_top_intro_tac;
  apply 0 (imp_elim_tac
    (mk_arrow_t (mk_var_t 0) mk_bot_t));
  apply 0 (assm_tac 0);
  apply 0 imp_intro_tac;
  apply 0 (mu_label_intro_tac 1);
  apply 0 (assm_tac 0);
qed ();
```

Extracted μML Program

$$\Lambda A.\ \neg\neg A \to A$$




```
tabs(A) ->
 fun (y : (A -> bot) -> bot) ->
  bind (a : A) ->
   [abort]. (y (fun (x : A) ->
                bind (b : bot) ->
                 [a]. x
                end
               end))
  end
 end
end

: forall(A)(((A -> bot) -> bot) -> A)
```

Conclusion

- Classical theorem prover with explicit evidence:
  - Extended $\lambda\mu$-calculus;
  - Evidence: μML terms;
  - Realisation: μTP.
- Future work:
  - Classical F_ω;
  - Extend μTP.