Formalisation of the $\lambda\mu^{\mathbf{T}}$-calculus in Isabelle/HOL

{ Cristina.Matache }@stx.ox.ac.uk

University of Cambridge

https://bitbucket.org/Cristina_Matache/prog-classical-types

The $\lambda\mu^{\mathbf{T}}$-calculus

The $\lambda\mu^{\mathbf{T}}$-calculus [2] is a combination between $\lambda\mu$ [3] and Gödel's System T:

$$\sigma, \tau ::= \mathtt{N} \mid \bot \mid \sigma \to \tau \qquad \text{types}$$

$$t, r, s ::= x \mid \lambda x{:}\sigma.t \mid t\ s \mid \mu\alpha{:}\sigma.c \mid 0 \mid \mathtt{S}\ t \mid \mathtt{nrec}_\sigma\ r\ s\ t \qquad \text{terms}$$

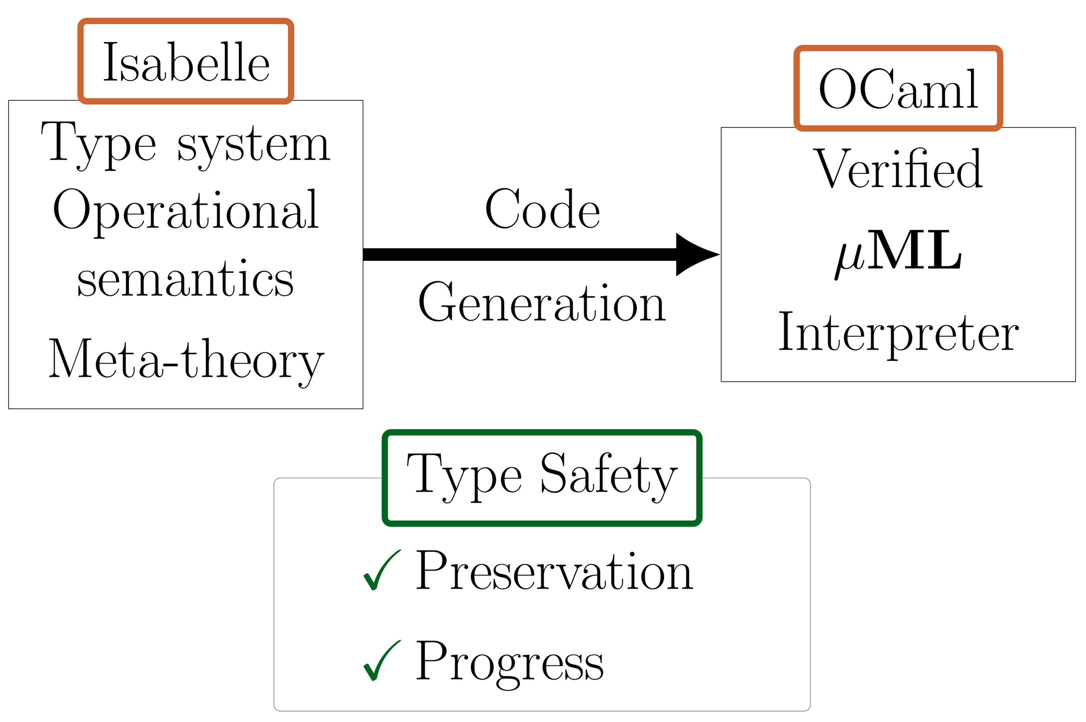
$$c ::= [\alpha]t \mid [\uparrow]t \qquad \text{named terms}$$

- $\mathtt{nrec}_\sigma\ r\ s\ t$—primitive recursion on the built-in natural numbers (from System T).
- $\mu\alpha{:}\sigma.c$—a new binding form; α is a μ-variable, disjoint from λ-variables; σ is the type of the whole expression.
- $\uparrow$—abort, a μ-constant; it behaves like a free μ-variable with respect to substitution; it can never be bound by a μ.
- In $\mu\alpha{:}\sigma.(\ldots[\alpha]t\ldots)$, α is a **channel** on which the value of t is transmitted, from $[\alpha]$ to $\mu\alpha{:}\sigma$.

Contributions

1. $\lambda\mu^{\mathbf{T}}$ formalisation using de Bruijn indices.
2. Mechanised Type Safety proofs.
3. Correspondence to *full* classical logic [1].
4. Datatypes in 'direct style': booleans, products and tagged unions.
5. μ**ML**: a prototype programming language with classical types, based on $\lambda\mu^{\mathbf{T}}$.
6. Verified interpreter for μ**ML**.

μML Interpreter



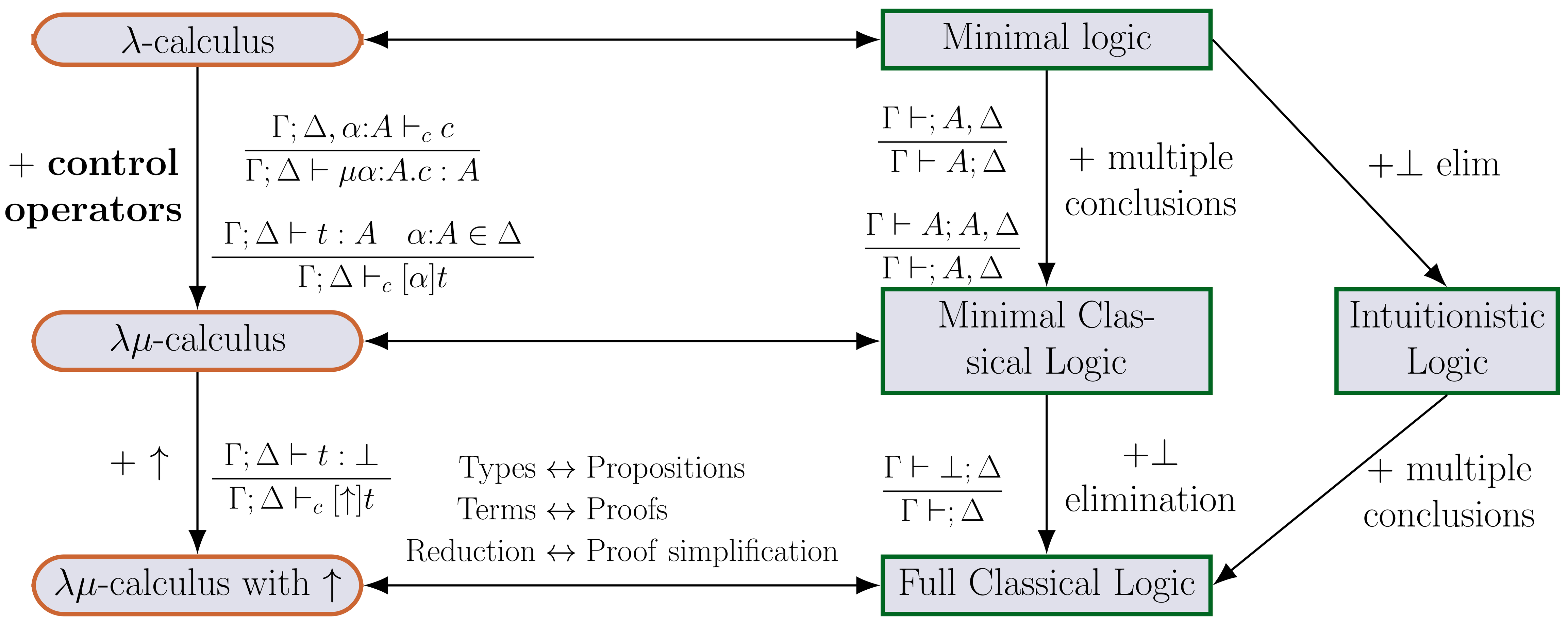
Potential μ**ML** applications:

1. Classical proof terms.
2. Proof exchange mechanism.
3. Expressive algorithmic representation (e.g. backtracking via control).

Work in progress:

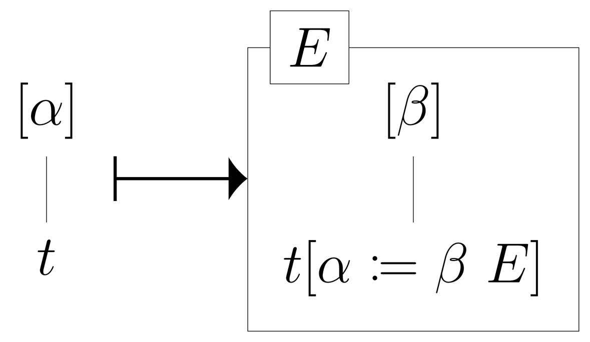
1. Extensions: higher-order polymorphism (F_ω)
2. Prototype theorem prover.

Propositions-as-Types



Beta-Reduction

Structural substitution, $s[\alpha := \beta\ E]$, acts on subterms of s labelled by (a free) α, as below:



Contexts:

$$E ::= \Box \mid E\ t \mid \mathtt{S}\ E \mid \mathtt{nrec}_\sigma\ r\ s\ E$$

$$([\alpha]t)[\alpha := \beta\ E] := [\beta]E[t[\alpha := \beta\ E]]$$

Reduction rules for μ and named terms:

$$(\mu\alpha{:}\sigma{\to}\tau.c)\ s \longrightarrow \mu\alpha{:}\tau.c[\alpha := \alpha\ (\Box\ s)] \quad (\mu R)$$

$$\mathtt{S}\ (\mu\alpha{:}\sigma.c) \longrightarrow \mu\alpha{:}\sigma.c[\alpha := \alpha\ (\mathtt{S}\ \Box)] \quad (\mu\mathtt{S})$$

$$\mathtt{nrec}_\tau\ r\ s\ (\mu\alpha{:}\sigma.c) \longrightarrow \mu\alpha{:}\tau.c[\alpha := \alpha\ (\mathtt{nrec}_\tau\ r\ s\ \Box)] \quad (\mu\mathtt{N})$$

$$\mu\alpha{:}\sigma.[\alpha]t \longrightarrow t \quad \text{if } \alpha \text{ not free in } t \quad (\mu\eta)$$

$$[\alpha]\mu\beta{:}\sigma.c \longrightarrow c[\beta := \alpha\ \Box] \quad (\mu i)$$

Control Operators

catch α **in** $t = \mu\alpha{:}\tau.[\alpha]t$

throw s **to** $\alpha = \mu\beta{:}\sigma.[\alpha]s \quad \beta \neq \alpha$, β not free in s

Example: **catch** α **in** ((**throw** 0 **to** α) (S 0))

$$\mu\alpha{:}\mathtt{N}.[\alpha]((\mu\beta{:}\mathtt{N}{\to}\mathtt{N}.[\alpha]0)\ (\mathtt{S}\ 0))$$

$$\longrightarrow_{(\mu R)} \mu\alpha{:}\mathtt{N}.[\alpha]\mu\beta{:}\mathtt{N}.([\alpha]0)[\beta := \beta\ (\Box\ (\mathtt{S}0))]$$

$$= \mu\alpha{:}\mathtt{N}.[\alpha](\mu\beta{:}\mathtt{N}.[\alpha]0) \quad \text{since } \beta \text{ not free in } [\alpha]0$$

$$\longrightarrow_{(\mu i)} \mu\alpha{:}\mathtt{N}.[\alpha]0$$

$$\longrightarrow_{(\mu\eta)} 0 \quad \text{since } \alpha \text{ is not free in } 0$$

This can be generalised to:

$$\textbf{catch}\ \alpha\ \textbf{in}\ (E[\textbf{throw}\ s\ \textbf{to}\ \alpha]) \longrightarrow^* s$$

References

[1] Z. M. Ariola and H. Herbelin. Minimal classical logic and control operators. In *ICALP*, 2003.

[2] H. Geuvers, R. Krebbers, and J. McKinna. The $\lambda\mu^{\mathbf{T}}$-calculus. *Ann. Pure Appl. Log.*, 164(6), 2013.

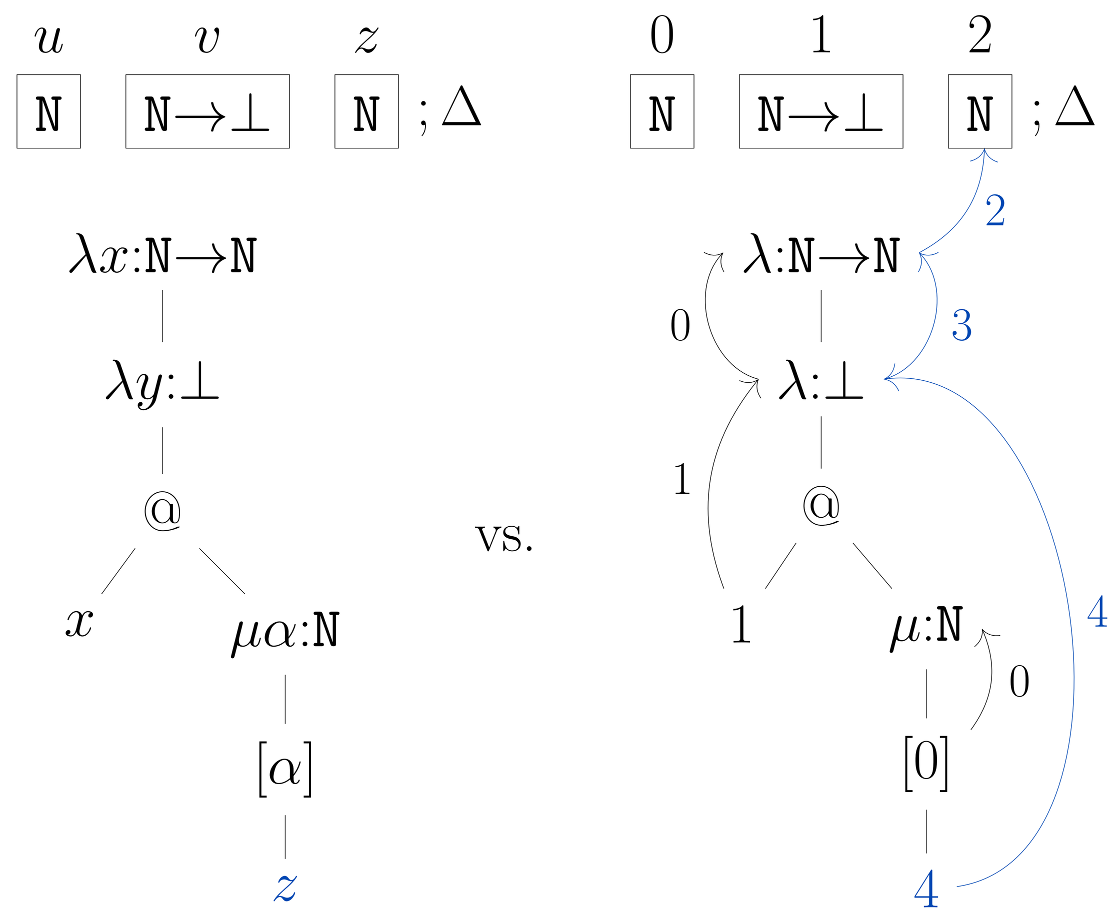
[3] M. Parigot. $\lambda\mu$-Calculus. In *LPAR*, 1992.

De Bruijn Notation

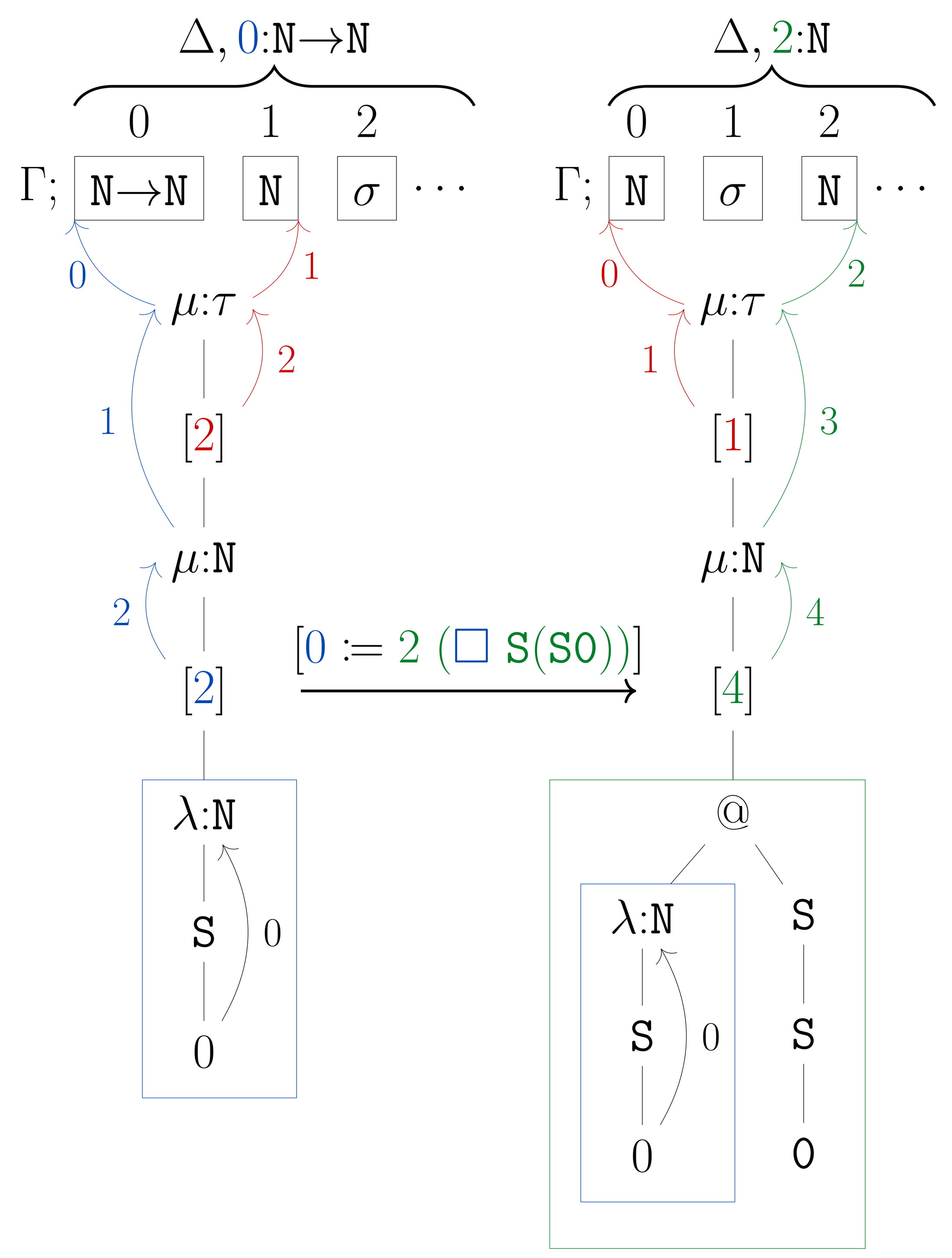
$$u{:}\mathtt{N}, v{:}\mathtt{N}{\to}\bot, z{:}\mathtt{N}; \Delta \vdash \lambda x{:}\mathtt{N}{\to}\mathtt{N}.\lambda y{:}\bot.(x\ \mu\alpha{:}\mathtt{N}.[\alpha]z)$$

vs.

$$0{:}\mathtt{N}, 1{:}\mathtt{N}{\to}\bot, 2{:}\mathtt{N}; \Delta \vdash \lambda{:}\mathtt{N}{\to}\mathtt{N}.\lambda{:}\bot.(1\ \mu{:}\mathtt{N}.[0]4)$$



De Bruijn indices of free μ-variables need to be carefully adjusted when performing structural substitution, in order to preserve typing:



Similarly, during reduction:

$$\Gamma; 1{:}\mathtt{N} \vdash \mu{:}\mathtt{N}.[0](\mu{:}\mathtt{N}.[3]0) \longrightarrow_{(\mu\eta)/(\mu i)} \Gamma; 1{:}\mathtt{N} \vdash \mu{:}\mathtt{N}.[2]0$$