An implementation of Cubical Type Theory

Mattia Furlan

25 aprile 2022

Indice

1 Syntax of the language

The syntax of the language is given by the following grammar, where A, B, M, K are meta-variables for terms and K represents (possibly) neutral terms.

$$
\begin{aligned}
A, B, M, K &::= \mathtt{U} \mid K \mid [x : A]M \\
&\quad \mid \mathtt{N} \mid \mathtt{0} \mid \mathtt{S} \\
&\quad \mid \mathbb{I} \mid \mathit{Sys} \mid \mathit{Partial} \mid \mathit{Restr} \\
K &::= x \mid KM \mid \mathtt{ind}\ MMMK \\
\mathit{Sys} &::= [\psi \mapsto M, \ldots, \psi \mapsto M] \\
\mathit{Partial} &::= [\varphi]A \\
\mathit{Restr} &::= [\psi \mapsto M, \ldots, \psi \mapsto M]A
\end{aligned}
$$

$$
\begin{aligned}
\chi &::= (i = 0) \mid (i = 1) \mid (i = j) \\
\psi &::= \chi \mid \chi \wedge \psi \\
\varphi &::= \psi \mid \psi \vee \varphi
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Program} &::= [\mathit{TopLevel}] \\
\mathit{TopLevel} &::= \mathit{Definition} \mid \mathit{Declaration} \mid \mathit{Example} \\
\mathit{Definition} &::= x : A = B \\
\mathit{Declaration} &::= x : A \\
\mathit{Example} &::= M
\end{aligned}
$$

In more detail, the language has symbols:

- `U` for the universe of types.[1]
- `N` for the type of natural numbers, with costants `0` and `S` respectively for zero and the successor operator. `ind` $F\ c_0\ c_s\ n$ is the induction principle for naturals (where $F : \mathtt{N} \to \mathtt{U}$ is a type family).
- $[x : A]M$ both for the λ and Π abstractions.
- $\mathbb{I}$ for the formal interval $[0, 1]$.
- Formulas. We use ψ for conjuctions of atomic formulas and φ for formulas in disjunctive normal form.

We also consider values, i.e. the results of the evaluation of terms, as terms extended with a closure operator of the form $\langle x : ty, M, \Gamma \rangle$, where the abstracted variable x is paired with its type ty, the abstraction body is M and the closure enviroment is specified by the context Γ (see next section). We write meta-variables representing values in bold, i.e. $\mathbf{v}$, to distinguish them from values.

The only change we make in the abstract syntax is that we annotate neutral values with their (evaluated) type, written as $\mathbf{v}^{\mathbf{t}}$.[2] Neutral values get annotated

[1] As of now, it holds (inconsistently) that `U` : `U`.

[2] In the implementation, we use a field of type `Maybe Value` for this, so that terms have this field equal to `Nothing`, while values have '`Just`' their type.

during evaluation, so the EVAL function needs the context to lookup the types; instead of having to give both the context and the enviroment, we use only the context, adding values bindings to it. Type annotations are required in order to simplify cubical expressions: for example if the variable x has type $[i : \mathbb{I}][(i = 0) \mapsto a, (i = 1) \mapsto b]A$, then $x\ i$ shall reduce to a under the binding $i \mapsto 0$.

2 Contexts and enviroments

The implementation uses two data structures to store information: a directions enviroment Θ, which is only used to store bindings of the form $i \mapsto 0$, $i \mapsto 1$, $i \mapsto j$, and a context, which stores definitions, declarations and values bindings. Evaluation only needs the context, whereas type-checking requires both the context and the evaluation enviroment.

Definition 2.1 (Context)**.**

1. $()$ *is the empty context;*
2. $\Gamma, x : A = B$ *extends the context* Γ *with a definition, where both the type* A *and the body* B *are terms;*
3. $\Gamma, x : A$ *extends the context* Γ *with a declaration, where the type* A *is a term;*
4. $\Gamma, x \mapsto \mathbf{t}$ *extends the context* Γ *binding the variable* x *to the value* $\mathbf{t}$*.*

Note that item (4) is used only during evalutation.

To manage and simplify formulas efficiently, we use a particular data structure called *directions enviroment*; this is formed by:

- A list *zeros* of names which shall be replaced by $\mathtt{0}_{\mathbb{I}}$.
- A list *ones* of names which shall be replaced by $\mathtt{1}_{\mathbb{I}}$.
- A list of partitions (called diagonals), that is, a list of lists of names; each partition represents the names which shall be identified.

Albeit it is implemented using the aforementioned lists, we will use a simple notation for the directions enviroment, as if it were a list of bindings.

Definition 2.2 (Directions enviroment)**.**

1. $()$ *is the empty directions enviroment;*
2. $\Theta, i \mapsto \mathtt{0}_{\mathbb{I}}$ *(or* $i \mapsto \mathtt{1}_{\mathbb{I}}$*) extends the directions enviroment* Θ *binding the name* i *to* $\mathtt{0}_{\mathbb{I}}$ *(or to* $\mathtt{1}_{\mathbb{I}}$*);*
3. $\Theta, i \mapsto j$ *extends the directions enviroment* Θ *binding the name* i *to the name* j *(diagonal).*

3 Evaluation

Given a context Γ, the function EVAL_Γ gets a term as input and gives the corrisponding value as output. Instead of writing $\text{EVAL}_\Gamma(t)$, we shall simply write

$(t)_\Gamma$. We denote with $(\varphi)_\Gamma$ the substitution in the formula φ of the bindings stored in Γ.[3]

$$
\begin{aligned}
(x)_\Gamma &= x^{\textsc{lookupType}(\Gamma,x)} \\
(\mathtt{U})_\Gamma &= \mathtt{U} \\
([x : ty]e)_\Gamma &= \langle x : ty, e, \Gamma\rangle \\
(f\ a)_\Gamma &= \textsc{app}((f)_\Gamma, (a)_\Gamma) \\
(\mathtt{N})_\Gamma &= \mathtt{N} \\
(\mathtt{0})_\Gamma &= \mathtt{0} \\
(\mathtt{S}\ n)_\Gamma &= \mathtt{S}\ (n)_\Gamma \\
(\mathtt{ind}\ F\ c_0\ c_s\ n)_\Gamma &= \textsc{ind}((F)_\Gamma, (c_0)_\Gamma, (c_s)_\Gamma, (n)_\Gamma) \\
(\mathbb{I})_\Gamma &= \mathbb{I} \\
([\psi_1 \mapsto M_1, \ldots, \psi_n \mapsto M_n])_\Gamma &= [(\psi_1)_\Gamma \mapsto (M_1)_\Gamma, \ldots, (\psi_n)_\Gamma \mapsto (M_n)_\Gamma] \\
([\varphi]A)_\Gamma &= \textsc{foldPartial}[(\varphi)_\Gamma](A)_\Gamma \\
([\psi_1 \mapsto M_1, \ldots, \psi_n \mapsto M_n]A)_\Gamma &= \textsc{foldRestr}\big([(\psi_1)_\Gamma \mapsto (M_1)_\Gamma, \ldots, (\psi_n)_\Gamma \mapsto (M_n)_\Gamma](A)_\Gamma\big)
\end{aligned}
$$

The evalutation function uses some helpers for the evaluation of (possibly) neutral terms:

- APP applied to **f** and **a** first checks if **f** is a (eventually restricted) closure; in that case it evaluates the closure, otherwise **f a** must be a neutral application which gets annotated with its type.

$$
\begin{aligned}
&\textsc{app}(\langle x : ty, e, \Gamma\rangle, \mathbf{a}) = (e)_{\Gamma, x\mapsto\mathbf{a}} \\
&\textsc{app}([\vec{\psi} \mapsto \vec{\mathbf{t}}]\mathbf{f}, \mathbf{a}) = \textsc{app}(\mathbf{f}, \mathbf{a}) \\
&\textsc{app}(\mathbf{f}^{\langle x:ty,e,\Gamma\rangle}, \mathbf{a}) = (\mathbf{f}\ \mathbf{a})^{\textsc{app}(\langle x:ty,e,\Gamma\rangle,\mathbf{a})}
\end{aligned}
$$

- IND evaluates an induction by pattern-matching on the argument n; if n is neutral, then the whole induction is neutral and so it gets annotated with its type.

$$
\begin{aligned}
&\textsc{ind}(\mathbf{ty}, \mathbf{c}_0, \mathbf{c}_s, \mathtt{0}) = \mathbf{c}_0 \\
&\textsc{ind}(\mathbf{ty}, \mathbf{c}_0, \mathbf{c}_s, \mathtt{S}\ \mathbf{m}) = \textsc{app}(\textsc{app}(\mathbf{c}_s, \mathbf{m}), \textsc{ind}(\mathbf{ty}, \mathbf{c}_0, \mathbf{c}_s, \mathbf{m})) \\
&\textsc{ind}(\mathbf{ty}, \mathbf{c}_0, \mathbf{c}_s, \mathbf{n}) = (\mathtt{ind}\ \mathbf{ty}\ \mathbf{c}_0\ \mathbf{c}_s\ \mathbf{n})^{\textsc{app}(\mathbf{ty},\mathbf{n})}
\end{aligned}
$$

The routine FOLDRESTR collapses all the terms or values the form $[\vec{\psi_1} \mapsto \vec{M_1}] \ldots [\vec{\psi_n} \mapsto \vec{M_n}]A$, where A is not a restriction type, to $[\vec{\psi_1} \mapsto \vec{M_1}, \ldots, \vec{\psi_n} \mapsto \vec{M_n}]A$; the routine FOLDPARTIAL works analogously. This simplifies the type-checker since we put restriction and partial types in a kind of normal form.

[3]Note that the only possible bindings for interval variables are renamings, e.g. of the form $i \mapsto j$. Interval variables renaming is required since, for example, given $q : \mathtt{Path}\ A\ b\ c \equiv [i : I][i = 0 \mapsto b, i = 1 \mapsto c]A$, then $q\ j$ should be annotated with the type $[j = 0 \mapsto b, j = 1 \mapsto c]A$.

4 Type checking/inference rules

The implementation uses a bidirectional type checker: I denote type inference with $e \Rightarrow \mathbf{t}$ (the input is a term a and the output is a value $\mathbf{t}$) and type checking with $e \Leftarrow \mathbf{t}$ (both are inputs, where the term e is checked against value $\mathbf{t}$). I use the following notations:

- $\Gamma \vdash \varphi : \mathbb{F}$ means that φ is a formula whose variables are declared in Γ.
- $\mathbf{t}_1 \sim^{\Theta}_{\tau(\Gamma)} \mathbf{t}_2$ means that $\mathbf{t}_1$ and $\mathbf{t}_2$ are α-equivalent under the contraints in Θ; $\tau(\Gamma)$ is the list of names declared in the context Γ.
- $\psi_1 \sim \psi_2$ means that ψ_1 and ψ_2 are logically equivalent formulas.

4.1 Type inference

$$\frac{}{\Gamma; \Theta \vdash x \Rightarrow \textsc{lookupType}(\Gamma, x)} \quad \text{(Var)}$$

$$\frac{}{\Gamma; \Theta \vdash \mathtt{U} \Rightarrow \mathtt{U}} \quad \text{(Universe)}$$

$$\frac{\Gamma; \Theta \vdash f \Rightarrow \langle s : ty, e, \Gamma' \rangle \quad \Gamma; \Theta \vdash a \Leftarrow (ty)_{\Gamma'}}{\Gamma; \Theta \vdash f\ a \Rightarrow \textsc{app}(\langle s : ty, e, \Gamma' \rangle, (a)_{\Gamma})} \quad \text{(App)}$$

$$\frac{}{\Gamma; \Theta \vdash \mathtt{N} \Rightarrow \mathtt{U}} \quad \text{(Nat)}$$

$$\frac{}{\Gamma; \Theta \vdash \mathtt{0} \Rightarrow \mathtt{N}} \quad \text{(Nat-Zero)}$$

$$\frac{\Gamma; \Theta \vdash n \Leftarrow \mathtt{N}}{\Gamma; \Theta \vdash \mathtt{S}\ n \Rightarrow \mathtt{N}} \quad \text{(Nat-Succ)}$$

$$\frac{\begin{array}{cc} \Gamma; \Theta \vdash F \Leftarrow (\mathtt{N} \to U)_{()} & \Gamma; \Theta \vdash c_0 \Leftarrow \textsc{app}((F)_{\Gamma}, \mathtt{0}) \\ \Gamma; \Theta \vdash n \Leftarrow \mathtt{N} & \Gamma; \Theta \vdash c_s \Leftarrow ([m : \mathtt{N}]Fm \to F(\mathtt{S}\ m))_{\Gamma} \end{array}}{\Gamma; \Theta \vdash \mathtt{ind}\ F\ c_0\ c_s\ n \Rightarrow \textsc{app}((F)_{\Gamma}, (n)_{\Gamma})} \quad \text{(Nat-Ind)}$$

$$\frac{}{\Gamma; \Theta \vdash \mathbb{I} \Rightarrow \mathtt{U}} \quad \text{(Interval)}$$

4.2 Type checking

$$\frac{\Gamma \vdash t \Leftarrow \mathtt{U} \quad \Gamma, x : t \vdash e \Leftarrow \mathtt{U}}{\Gamma \vdash [x : t]e \Leftarrow \mathtt{U}} \tag{Pi}$$

$$\frac{\begin{array}{c}\Gamma;\Theta \vdash ty \Leftarrow \mathtt{U} \qquad (ty)_\Gamma \sim_{\tau(\Gamma)} (ty_1)_{\Gamma_1} \\ \Gamma, x : ty, x \mapsto v; \Theta \vdash e \Leftarrow \textsc{app}(\langle x_1 : ty, e_1, \Gamma_1\rangle, v^{(ty_1)_{\Gamma_1}})\end{array}}{\Gamma;\Theta \vdash [x : ty]e \Leftarrow \langle x_1 : ty, e_1, \Gamma_1\rangle} \; (v = \textsc{newVar}(\Gamma_1, x)) \tag{Lambda}$$

$$\frac{\Gamma \vdash \psi_i : \mathbb{F} \quad (\psi_1 \vee \cdots \vee \psi_n) \sim \varphi \quad \Gamma;\Theta, \psi_i \vdash t_i \Leftarrow \mathbf{A} \quad (t_i)_\Gamma \sim^{\Theta, \psi_i \wedge \psi_j}_{\tau(\Gamma)} (t_j)_\Gamma \; (1 \le i, j \le n)}{\Gamma \vdash [\psi_1 \mapsto t_1, \ldots, \psi_n \mapsto t_n] \Leftarrow [\varphi]\mathbf{A}} \; (A \neq \mathbb{I}) \tag{System}$$

$$\frac{\Gamma \vdash \psi_i : \mathbb{F} \quad \Gamma;\Theta, \psi_i \vdash A \Leftarrow \mathtt{U} \; (1 \le i \le n)}{\Gamma;\Theta \vdash [\psi_1 \vee \cdots \vee \psi_n]A \Leftarrow \mathtt{U} \quad (A \neq \mathbb{I})} \tag{Partial}$$

$$\frac{\Gamma;\Theta \vdash A \Leftarrow \mathtt{U} \quad \Gamma \vdash \psi_i : \mathbb{F} \quad \Gamma;\Theta, \psi_i \vdash t_i \Leftarrow (A)_\Gamma \; (1 \le i \le n)}{\Gamma;\Theta \vdash [\psi_1 \mapsto t_1, \ldots, \psi_n \mapsto t_n]A \Leftarrow \mathtt{U}} \; (A \neq \mathbb{I}) \tag{Restr-U}$$

$$\frac{\Gamma;\Theta \vdash e \Leftarrow \mathbf{A} \quad (e)_\Gamma \sim^{\Theta, \psi_i}_{\tau(\Gamma)} \mathbf{t}_i}{\Gamma;\Theta \vdash e \Leftarrow [\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]\mathbf{A}} \; (A \neq \mathbb{I}) \tag{Restr}$$

$$\frac{\Gamma \vdash e \Rightarrow [\psi_2]\mathbf{A}_1 \quad \psi_1 \le \psi_2 \quad \mathbf{A}_1 \sim^{\Theta}_{\tau(\Gamma)} \mathbf{A}_2}{\Gamma \vdash e \Leftarrow [\psi_1]\mathbf{A}_2} \tag{Weak-Partial}$$

$$\frac{\Gamma \vdash e \Rightarrow [\psi_2]\mathbf{A}_1 \quad \psi_1 \le \psi_2 \quad \mathbf{A}_1 \sim^{\Theta}_{\tau(\Gamma)} \mathbf{A}_2}{\Gamma \vdash e \Leftarrow [\psi_1]\mathbf{A}_2} \tag{Weak-Restr}$$

$$\frac{\Gamma \vdash e \Rightarrow \mathbf{A}_1 \quad \mathbf{A}_1 \sim^{\Theta}_{\tau(\Gamma)} \mathbf{A}_2}{\Gamma \vdash e \Leftarrow \mathbf{A}_2} \tag{Infer}$$

4.3 α-conversion

The function $\cdot \sim^{\Theta}_{ns} \cdot$ tests α-conversion between two values, given a directions enviroment Θ; it keeps track of the list of already used names, to avoid conflicts when reading-back closures.

$$\frac{}{\mathtt{U} \sim^{\Theta}_{ns} \mathtt{U}}$$

$$\frac{}{\mathbf{x} \sim^{\Theta}_{ns} \mathbf{x}}$$

$$\frac{(t)_{\Gamma} \sim^{\Theta}_{ns} (t')_{\Gamma'} \quad (e)_{\Gamma, x:t, x\mapsto v} \sim^{\Theta}_{ns,v} (e')_{\Gamma', y:t, y\mapsto v}}{\langle x : ty, e, \Gamma\rangle \sim^{\Theta}_{ns} \langle y : ty', e', \Gamma'\rangle} \ (v = \textsc{newVar}(\Gamma', x))$$

$$\frac{\mathbf{f} \sim^{\Theta}_{ns} \mathbf{f}' \quad \mathbf{a} \sim^{\Theta}_{ns} \mathbf{a}'}{\mathbf{f}\ \mathbf{a} \sim^{\Theta}_{ns} \mathbf{f}'\ \mathbf{a}'}$$

$$\frac{\psi_i \sim \mathtt{True} \quad \mathbf{t}_i \sim^{\Theta}_{ns} \mathbf{v}}{(\mathbf{f}\ \mathbf{i})^{[\psi_1\mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]\mathbf{A}} \sim^{\Theta}_{ns} \mathbf{v}}$$

$$\frac{\psi_i \sim \mathtt{True} \quad \mathbf{v} \sim^{\Theta}_{ns} \mathbf{t}_i}{\mathbf{v} \sim^{\Theta}_{ns} (\mathbf{f}\ \mathbf{i})^{[\psi_1\mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]\mathbf{A}}}$$

$$\frac{}{\mathtt{N} \sim^{\Theta}_{ns} \mathtt{N}}$$

$$\frac{}{\mathtt{0} \sim^{\Theta}_{ns} \mathtt{0}}$$

$$\frac{\mathbf{n} \sim^{\Theta}_{ns} \mathbf{n}'}{\mathtt{S}\ \mathbf{n} \sim^{\Theta}_{ns} \mathtt{S}\ \mathbf{n}'}$$

$$\frac{}{\mathbb{I} \sim^{\Theta}_{ns} \mathbb{I}}$$

$$\frac{\mathbf{F} \sim^{\Theta}_{ns} \mathbf{F}' \quad \mathbf{c}_0 \sim^{\Theta}_{ns} \mathbf{c}'_0 \quad \mathbf{c}_s \sim^{\Theta}_{ns} \mathbf{c}'_s \quad \mathbf{n} \sim^{\Theta}_{ns} \mathbf{n}'}{\mathtt{ind}\ \mathbf{F}\ \mathbf{c}_0\ \mathbf{c}_s\ \mathbf{n} \sim^{\Theta}_{ns} \mathtt{ind}\ \mathbf{F}'\ \mathbf{c}'_0\ \mathbf{c}'_s\ \mathbf{n}'}$$

$$\frac{(\psi_1 \vee \cdots \vee \psi_n) \sim (\psi'_1 \vee \cdots \vee \psi'_n) \quad \mathbf{t}_i \sim^{\Theta, \psi_i \wedge \psi'_j}_{ns} \mathbf{t}'_j}{[\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n] \sim^{\Theta}_{ns} [\psi'_1 \mapsto \mathbf{t}'_1, \ldots, \psi'_m \mapsto \mathbf{t}'_m]} \ (1 \le i \le n, 1 \le j \le m)$$

$$\frac{\psi_i \sim \texttt{True} \quad \mathbf{t}_i \sim^{\Theta}_{ns} \mathbf{v}}{[\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n] \sim^{\Theta}_{ns} \mathbf{v}}$$

$$\frac{\psi_i \sim \texttt{True} \quad \mathbf{v} \sim^{\Theta}_{ns} \mathbf{t}_i}{\mathbf{v} \sim^{\Theta}_{ns} [\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]}$$

$$\frac{\varphi \sim \varphi' \quad \mathbf{A} \sim^{\Theta}_{ns} \mathbf{A}'}{[\varphi]\mathbf{A} \sim^{\Theta}_{ns} [\varphi']\mathbf{A}'}$$

$$\frac{[\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n] \sim^{\Theta}_{ns} [\psi'_1 \mapsto \mathbf{t}'_1, \ldots, \psi'_m \mapsto \mathbf{t}'_m] \quad \mathbf{A} \sim^{\Theta}_{ns} \mathbf{A}'}{[\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]\mathbf{A} \sim^{\Theta}_{ns} [\psi'_1 \mapsto \mathbf{t}'_1, \ldots, \psi'_m \mapsto \mathbf{t}'_m]\mathbf{A}'}$$