An implementation of Cubical Type Theory

Mattia Furlan

07/07/2022

Contents

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.

$$
\begin{aligned}
A, B, M, K ::= {} & \mathtt{U} \mid K \mid [x : A]M \\
& \mid \langle x : A \rangle M \mid (M, M) \mid M.1 \mid M.2 \\
& \mid A + B \mid \mathtt{inl}\ M \mid \mathtt{inr}\ M \mid \mathtt{split}\ A\ M\ M\ M \\
& \mid \mathtt{N} \mid \mathtt{0} \mid \mathtt{S} \\
& \mid \mathbb{I} \mid \mathtt{0} \mid \mathtt{1} \\
& \mid \mathit{Sys} \mid \mathit{Partial} \mid \mathit{Restr} \mid \mathit{Comp} \\
K ::= {} & x \mid KM \mid \mathtt{ind}\ MMMK \\
\mathit{Sys} ::= {} & [\psi \mapsto M, \ldots, \psi \mapsto M] \\
\mathit{Partial} ::= {} & [\varphi]A \\
\mathit{Restr} ::= {} & [\psi \mapsto M, \ldots, \psi \mapsto M]A \\
\mathit{Comp} ::= {} & \mathtt{comp}\ M\ \varphi\ M\ M\ M\ M
\end{aligned}
$$

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

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

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 is a type family).
- $[x : A]M$ both for the λ and Π abstractions; when x does not occur in M, we shall simply write $A \to M$.
- $\langle x : A \rangle M$ for the dependent sum, with the pair constructor $(\cdot, \cdot)$ and the projections .1, .2; when x does not occur in M, we shall simply write $A * M$.
- $A + B$ for the coproduct type, with injections `inl` and `inr`; the destructor is `split` $A\ t\ f_1\ f_2$.
- $\mathbb{I}$ for the formal interval $[0, 1]$. The endpoints are `0` and `1`.

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

- 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 $\textsc{closure}(M, \Gamma)$, where M is an (unevaluated) term of the form $[x : A]B$ or $\langle x : A\rangle B$ 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 terms.

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 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 $i \mapsto 0$.

1.1 Syntax for composition

We use the following notation for compositions: $\texttt{comp}\ F\ \varphi\ i_0\ u\ b\ i$, where

- $F : \mathbb{I} \to \texttt{U}$ is a type family;
- φ is the formula describing the sides of the composition figure;
- $i_0 : \mathbb{I}$ is the starting point of the composition;
- $u : [x : \mathbb{I}][\varphi]F\ x$ is the partial term, defined on φ, which will be extended by the composition;
- $b : [\varphi \mapsto u\ i_0]F\ i_0$ is the base case, defined on the starting point i_0; the term b must agree with $u\ i_0$ on φ;
- $i : \mathbb{I}$ is the end point of the composition.

The composition has type $[i = i_0 \mapsto b \mid \varphi \mapsto u\ i]F\ i$, i.e. it extends b on $i = i_0$ and u on φ.

The inference rule for composition is therefore the following:

$$\frac{\Gamma; \Theta \vdash F \Leftarrow (\mathbb{I} \to \texttt{U}) \quad \Gamma \vdash \varphi : \mathbb{F} \quad \Gamma; \Theta \vdash i_0 \Leftarrow \mathbb{I} \quad \Gamma; \Theta \vdash u \Leftarrow ([x : \mathbb{I}][\varphi]F\ x)_\Gamma \quad \Gamma; \Theta \vdash b \Leftarrow ([\varphi \mapsto u\ i_0]F\ i_0)_\Gamma \quad \Gamma; \Theta \vdash i \Leftarrow \mathbb{I}}{\Gamma; \Theta \vdash \texttt{comp}\ F\ \varphi\ i_0\ u\ b\ i \Rightarrow ([i = i_0 \mapsto b \mid \varphi \mapsto u\ i]F\ i)_\Gamma}$$

We make an example to make clear how composition works, demonstrating how to obtain the concatenation of two paths.

Suppose we are given a type $A : \texttt{U}$ and three points $a, b, c : A$, with paths $p : \texttt{Path}_A(a, b)$ and $q : \texttt{Path}_A(b, c)$; we want to obtain a path $pq : \texttt{Path}_A(a, c)$.

We draw a square having p as the base, so $i_0 \equiv \texttt{0}$ is the starting point and $p\ i$ is the base case, with the constant path $\texttt{refl}_A(a)$ as the left side and the path q as the right side. With composition (at the end point $\texttt{1}$) we finally obtain a path from a to c. In more detail, keeping in mind that we quantify over $i : \mathbb{I}$ before the composition,

[2] In the implementation, we use a field `Neutral Value Value` for annotated values.

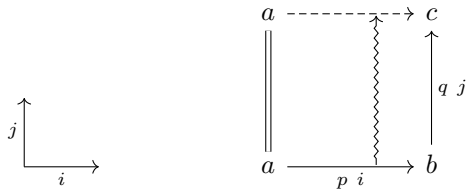

Figure 1: Concatenation of two paths. The result is the dotted line (lid of the square), which is a path connecting a to c. The twisted line only represents the act of composition (along the direction j), obtaining a point on the lid from the base point $p\ i$.

- F is the constant type family $I \to A$;
- φ is $i = 0 \vee i = 1$ (left and right sides of the square);
- i_0 is the starting point 0;
- u is the partial term defined only on the sides, given by $[j : \mathbb{I}][i = 0 \mapsto a \mid i = 1 \mapsto q\ j]$;[3]
- b is the base case given by $p\ i$;
- The end point is 1.[4]

The code for concatenation would therefore be:

```
concat : [A:U][a,b,c:A] Path A a b -> Path A b c -> Path A a c
 = [A:U][a,b,c:A][p:Path A a b][q:Path A b c][i:I]
   comp (const A) (i = 0 \/ i = 1) 0 ([j:I][i = 0 -> a | i = 1 -> q j]) (p i) 1 ;
```

where

```
Path : [A : U] A -> A -> U
 = [A : U][a0 : A][a1 : A][i : I][i = 0 -> a0 | i = 1 -> a1] A ;

refl : [A : U][a : A] Path A a a = [A : U][a : A][i : I] a ;

const : [A:U] I -> U = [A:U][i:I]A ;
```

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.

[3] We could also have written $\texttt{refl}_A(a)\ j$ instead of a in the system.

[4] We use 1 because we are only interested in the lid, not on the whole square (filler), which we would get by using i as the end point.

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 `0`.
- A list *ones* of names which shall be replaced by `1`.
- 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. $\Theta, i \mapsto \mathtt{0}$ *(or* $i \mapsto \mathtt{1}$*) extends the directions enviroment* Θ *binding the name* i *to* `0` *(or to* `1`*);*
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 $\textsc{eval}_\Gamma$ gets a term as input and gives the corrisponding value as output. Instead of writing $\textsc{eval}_\Gamma(t)$, we shall simply write $(t)_\Gamma$. We denote with $(\varphi)_\Gamma$ the substitution in the formula φ of the bindings stored in Γ.[5]

[5]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 $[j = 0 \mapsto b, j = 1 \mapsto c]A$.

$$
\begin{aligned}
(x)_\Gamma &= x^{\textsc{lookupType}(\Gamma,x)} \\
(\texttt{U})_\Gamma &= \texttt{U} \\
([x : ty]e)_\Gamma &= \textsc{closure}([x : ty]e, \Gamma) \\
(f\ a)_\Gamma &= \textsc{app}((f)_\Gamma, (a)_\Gamma) \\
(\langle x : ty \rangle e)_\Gamma &= \textsc{closure}(\langle x : ty \rangle e, \Gamma) \\
(t_1, t_2)_\Gamma &= ((t_1)_\Gamma, (t_2)_\Gamma) \\
(t.1)_\Gamma &= \textsc{fst}((t)_\Gamma) \\
(t.2)_\Gamma &= \textsc{snd}((t)_\Gamma) \\
(A + B)_\Gamma &= (A)_\Gamma + (B)_\Gamma \\
(\texttt{inl}\ t_1)_\Gamma &= \texttt{inl}\ (t_1)_\Gamma \\
(\texttt{inr}\ t_2)_\Gamma &= \texttt{inr}\ (t_2)_\Gamma \\
(\texttt{split}\ t\ ty\ f_1\ f_2)_\Gamma &= \textsc{split}((t)_\Gamma, (ty)_\Gamma, (f_1)_\Gamma, (f_2)_\Gamma) \\
(\texttt{N})_\Gamma &= \texttt{N} \\
(\texttt{Z})_\Gamma &= \texttt{Z} \\
(\texttt{S}\ n)_\Gamma &= \texttt{S}\ (n)_\Gamma \\
(\texttt{ind}\ F\ c_0\ c_s\ n)_\Gamma &= \textsc{ind}((F)_\Gamma, (c_0)_\Gamma, (c_s)_\Gamma, (n)_\Gamma) \\
(\mathbb{I})_\Gamma &= \mathbb{I} \\
(\texttt{0})_\Gamma &= \texttt{0} \\
(\texttt{1})_\Gamma &= \texttt{1} \\
([\psi_1 \mapsto M_1, \ldots, \psi_n \mapsto M_n])_\Gamma &= \textsc{simplify}_{()}\big([(\psi_1)_\Gamma \mapsto (M_1)_\Gamma, \ldots, (\psi_n)_\Gamma \mapsto (M_n)_\Gamma]\big) \\
([\varphi]A)_\Gamma &= \textsc{foldPartial}\big([(\varphi)_\Gamma](A)_\Gamma\big) \\
([\psi_1 \mapsto M_1, \ldots, \psi_n \mapsto M_n]A)_\Gamma &= \textsc{foldRestr}\big([(\psi_1)_\Gamma \mapsto (M_1)_\Gamma, \ldots, (\psi_n)_\Gamma \mapsto (M_n)_\Gamma](A)_\Gamma\big) \\
(\texttt{comp}\ F\ \varphi\ i_0\ u\ b)_\Gamma &= \textsc{comp}_\Gamma\big(F, \varphi, i_0, u, b\big)
\end{aligned}
$$

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, unless the type is a restriction type with a true formula, in which case the whole value gets reduced to the value given by the said formula.

  $$\textsc{app}(\textsc{closure}([x : ty]e, \Gamma), \mathbf{a}) = (e)_{\Gamma, x \mapsto \mathbf{a}}$$

  $$\textsc{app}([\vec{\psi} \mapsto \vec{\mathbf{t}}]\mathbf{f}, \mathbf{a}) = \textsc{app}(\mathbf{f}, \mathbf{a})$$

  $$\textsc{app}(\mathbf{f}^{\textsc{closure}([x:ty]e,\Gamma)}, \mathbf{a}) = \ \textit{case}\ \textsc{app}(\textsc{closure}([x : ty]e, \Gamma), \mathbf{a})\ \textit{of}$$

  $$\begin{cases} [\psi_1 \mapsto \mathbf{v_1}, \ldots, \psi_n \mapsto \mathbf{v_n}]\mathbf{A}, \textit{with}\ \psi_i\ \textit{true} & \longrightarrow \mathbf{v}_i \\ \textit{otherwise} & \longrightarrow (\mathbf{f}\ \mathbf{a})^{\textsc{app}(\textsc{closure}([x:ty]e,\Gamma),\mathbf{a})} \end{cases}$$

- 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.

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

- FST and SND return respectively the first and the second component of a pair, if the value is not neutral; otherwise it annotates the whole value with the corresponding type.

$$\begin{aligned}
&\textsc{fst}((\mathbf{v}_1, \mathbf{v}_2)) = \mathbf{v}_1\\
&\textsc{fst}(\mathbf{v}^{\textsc{closure}(\langle x:ty\rangle e,\Gamma)}) = \textsc{fst}(\mathbf{v})^{(ty)_\Gamma}\\
&\textsc{fst}(\mathbf{v}^{[\vec{\psi}\mapsto\vec{\mathbf{v}}]\mathbf{ty}}) = \textsc{fst}(\mathbf{v}^{\mathbf{ty}})\\
&\textsc{snd}((\mathbf{v}_1, \mathbf{v}_2)) = \mathbf{v}_2\\
&\textsc{snd}(\mathbf{v}^{\textsc{closure}(\langle x:ty\rangle e,\Gamma)}) = \textsc{snd}(\mathbf{v})^{(e)_{\Gamma,x\mapsto\mathbf{v}}}\\
&\textsc{snd}(\mathbf{v}^{[\vec{\psi}\mapsto\vec{\mathbf{v}}]\mathbf{ty}}) = \textsc{snd}(\mathbf{v}^{\mathbf{ty}})
\end{aligned}$$

- SPLIT does by-case analysis on the given argument, if not neutral, otherwise the whole term becomes neutral.

$$\begin{aligned}
&\textsc{split}(\mathtt{inl}\ \mathbf{v}_1, \mathbf{ty}, \mathbf{f}_1, \mathbf{f}_2) = \textsc{app}(\mathbf{f}_1, \mathbf{v}_1)\\
&\textsc{split}(\mathtt{inr}\ \mathbf{v}_2, \mathbf{ty}, \mathbf{f}_1, \mathbf{f}_2) = \textsc{app}(\mathbf{f}_2, \mathbf{v}_2)\\
&\textsc{split}(\mathbf{v}^{\mathbf{ty}'}, \mathbf{ty}, \mathbf{f}_1, \mathbf{f}_2) = (\mathtt{split}\ \mathbf{v}\ \mathbf{ty}\ \mathbf{f}_1\ \mathbf{f}_2)^{\textsc{app}(\mathbf{ty},\mathbf{v})}
\end{aligned}$$

- SIMPLIFY applied to a system returns the value corresponding to a true formula, if there is one, otherwise it does nothing.

$$\begin{aligned}
&\textsc{simplify}([\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]) = \mathbf{t}_i,\ \textit{if } \psi_i \textit{ is true}\\
&\textsc{simplify}(\mathbf{v}) = \mathbf{v},\ \textit{otherwise}
\end{aligned}$$

- $\textsc{comp}_\Gamma$ first checks if $(\varphi)_\Gamma$ is true; if so, it returns $(u\ i)_\Gamma$, otherwise it checks if $(i)_\Gamma$ is equal to $(i_0)_\Gamma$ (i.e. $(i)_\Gamma \sim^{\Theta}_{\tau(\Gamma)} (i_0)_\Gamma$), in which case it returns $(b)_\Gamma$. Otherwise, given a fresh variable v, it evaluates $F\ v$ to match against one of the following:
  - A product type $[x : ty]e$ (inside a closure with context Γ'). The result of the composition is a function (so, a closure) that given $y_i : ty(v/i)$,[6] returns
    $$\mathtt{comp}\ ([v : \mathbb{I}]ty')\ \varphi\ i_0\ u'\ (b\ \tilde{y}_0)\ i$$
    where
    $$\begin{aligned}
    \tilde{y}_0 &\equiv \mathtt{comp}\ (([v : \mathbb{I}]ty_1)\ \mathtt{False}\ i\ (\mathbb{I} \to [\,])\ y_i\ i_0\\
    \tilde{y} &\equiv \mathtt{comp}\ (([v : \mathbb{I}]ty_1)\ \mathtt{False}\ i\ (\mathbb{I} \to [\,])\ y_i\ v\\
    ty_1 &\equiv \textsc{readback}_{\tau(\Gamma)}((ty)_{\Gamma'})\\
    ty' &\equiv \textsc{readback}_{\tau(\Gamma)}\left((e)_{\Gamma',x\mapsto(\tilde{y})_\Gamma}\right)
    \end{aligned}$$
    The term u' is obtained by applying $\tilde{y}$ to u inside the closure.

[6] I denote with $ty(v/i)$ the result of substituing v to i inside ty.

– A sum type $\langle x : ty \rangle e$ (inside a closure with context Γ'). The result is a pair (c_0, c_1), where c_i is the composition of the i-argument, on the appropriate type family:

$$c_0 \equiv \textsc{comp}_\Gamma(([v : \mathbb{I}]ty_0), \varphi, i_0, u_0, (b.1), i)$$
$$c_1 \equiv \textsc{comp}_\Gamma(([v : \mathbb{I}]ty_1), \varphi, i_0, u_1, (b.2), i)$$

where

$$ty_0 \equiv \textsc{readback}_{\tau(\Gamma)}((ty)_{\Gamma'})$$
$$ty_1 \equiv \textsc{readback}_{\tau(\Gamma)}\left((e)_{\Gamma', x \mapsto c_0'}\right)$$
$$c_0' \equiv \textsc{comp}_\Gamma(([v : \mathbb{I}]ty_0), \varphi, i_0, u_0, (b.1), v)_\Gamma$$

The terms u_0 and u_1 are given respectively by applying the first and second projection to u inside the closure.

– Otherwise, if the type family is a neutral term, the whole composition is neutral, with type $[i = i_0 \mapsto b \mid \varphi \mapsto u\ i]F\ i$.

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.

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 $\alpha\eta$-equivalent modulo the contraints in Θ; $\tau(\Gamma)$ is the list of names declared in the context Γ.
- $\psi_1 \leq^{\Theta} \psi_2$ means that ψ_1 logically implies ψ_2 modulo the contraints in Θ.
- $\psi_1 \sim^{\Theta} \psi_2$ means that ψ_1 and ψ_2 are logically equivalent modulo the contraints in Θ, i.e. that $\psi_1 \leq^{\Theta} \psi_2$ and $\psi_2 \leq^{\Theta} \psi_1$.

We allow only a very restricted form of subtyping, specified by the rules (Weak-Partial) and (Weak-Restr).

4.1 Type inference

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

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

$$\frac{\Gamma;\Theta \vdash f \Rightarrow \textsc{closure}([x : ty]e, \Gamma') \quad \Gamma;\Theta \vdash a \Leftarrow (ty)_{\Gamma'}}{\Gamma;\Theta \vdash f\ a \Rightarrow \textsc{app}(\textsc{closure}([x : ty]e, \Gamma'), (a)_\Gamma)} \qquad \text{(App)}$$

$$\frac{\Gamma;\Theta \vdash f \Rightarrow [\psi_1 \mapsto \mathbf{g}_1, \ldots, \psi_n \mapsto \mathbf{g}_n]\textsc{closure}([x : ty]e, \Gamma') \quad \Gamma;\Theta \vdash a \Leftarrow (ty)_{\Gamma'}}{\Gamma;\Theta \vdash f\ a \Rightarrow [\psi_1 \mapsto \textsc{app}(\mathbf{g}_1, (a)_\Gamma), \ldots, \psi_n \mapsto \textsc{app}(\mathbf{g}_n, (a)_\Gamma)]\textsc{app}(\textsc{closure}([x : ty]e, \Gamma'), (a)_\Gamma)} \text{(App-Restr)}$$

$$\frac{\Gamma;\Theta \vdash p \Rightarrow \textsc{closure}(\langle x : ty\rangle e, \Gamma')}{\Gamma;\Theta \vdash p.1 \Rightarrow (ty)_{\Gamma'}} \qquad \text{(Sigma-1)}$$

$$\frac{\Gamma;\Theta \vdash p \Rightarrow \textsc{closure}(\langle x : ty\rangle e, \Gamma')}{\Gamma;\Theta \vdash p.2 \Rightarrow (e)_{\Gamma', x \mapsto (p.1)_\Gamma}} \qquad \text{(Sigma-2)}$$

$$\frac{\Gamma;\Theta \vdash p \Rightarrow [\psi_1 \mapsto \mathbf{q}_1, \ldots, \psi_n \mapsto \mathbf{q}_n]\textsc{closure}(\langle x : ty\rangle e, \Gamma')}{\Gamma;\Theta \vdash p.1 \Rightarrow [\psi_1 \mapsto \textsc{fst}(\mathbf{q}_1), \ldots, \psi_n \mapsto \textsc{fst}(\mathbf{q}_n)](ty)_{\Gamma'}} \qquad \text{(Sigma-1-Restr)}$$

$$\frac{\Gamma;\Theta \vdash p \Rightarrow [\psi_1 \mapsto \mathbf{q}_1, \ldots, \psi_n \mapsto \mathbf{q}_n]\textsc{closure}(\langle x : ty\rangle e, \Gamma')}{\Gamma;\Theta \vdash p.2 \Rightarrow [\psi_1 \mapsto \textsc{snd}(\mathbf{q}_1), \ldots, \psi_n \mapsto \textsc{snd}(\mathbf{q}_n)](e)_{\Gamma', x \mapsto (p.1)_\Gamma}} \text{(Sigma-2-Restr)}$$

$$\frac{\Gamma;\Theta \vdash t \Rightarrow \mathbf{ty_1} + \mathbf{ty_2} \quad \Gamma;\Theta \vdash f_1 \Leftarrow ([x : ty_1]ty\ (\texttt{inl}\ x))_\Gamma \quad \Gamma;\Theta \vdash f_2 \Leftarrow ([x : ty_2]ty\ (\texttt{inr}\ x))_\Gamma}{\Gamma;\Theta \vdash \texttt{split}\ t\ ty\ f_1\ f_2 \Rightarrow (ty\ t)_\Gamma} \text{(Sum-Split)}$$

$$\frac{\Gamma;\Theta \vdash F \Leftarrow (\mathbb{I} \to \mathtt{U}) \quad \Gamma \vdash \varphi : \mathbb{F} \quad \Gamma;\Theta \vdash i_0 \Leftarrow \mathbb{I} \quad \Gamma;\Theta \vdash u \Leftarrow ([x : \mathbb{I}][\varphi]F\ x)_\Gamma \quad \Gamma;\Theta \vdash b \Leftarrow ([\varphi \mapsto u\ i_0]F\ i_0)_\Gamma \quad \Gamma;\Theta \vdash i \Leftarrow \mathbb{I}}{\Gamma;\Theta \vdash \texttt{comp}\ F\ \varphi\ i_0\ u\ b\ i \Rightarrow ([i = i_0 \mapsto b \mid \varphi \mapsto u\ i]F\ i)_\Gamma} \text{(Comp)}$$

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

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

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

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

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

$$\frac{}{\Gamma;\Theta \vdash \mathtt{0} \Rightarrow \mathbb{I}} \quad \text{(Interval-0)}$$

$$\frac{}{\Gamma;\Theta \vdash \mathtt{1} \Rightarrow \mathbb{I}} \quad \text{(Interval-1)}$$

4.2 Type checking

$$\frac{\Gamma \vdash ty \Leftarrow \mathtt{U} \quad \Gamma, x : ty \vdash e \Leftarrow \mathtt{U}}{\Gamma \vdash [x : ty]e \Leftarrow \mathtt{U}} \quad \text{(Pi)}$$

$$\frac{\Gamma \vdash ty \Leftarrow \mathtt{U} \quad \Gamma, x : ty \vdash e \Leftarrow \mathtt{U}}{\Gamma \vdash \langle x : ty\rangle e \Leftarrow \mathtt{U}} \quad \text{(Sigma)}$$

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

$$\frac{\Gamma;\Theta \vdash s \Leftarrow (ty)_{\Gamma'} \qquad \Gamma;\Theta \vdash t \Leftarrow (e)_{\Gamma', x \mapsto (s)_\Gamma}}{\Gamma;\Theta \vdash (s, t) \Leftarrow \textsc{closure}(\langle x : ty\rangle e, \Gamma')} \quad \text{(Pair)}$$

$$\frac{\Gamma;\Theta \vdash A \Leftarrow \mathtt{U} \qquad \Gamma;\Theta \vdash B \Leftarrow \mathtt{U}}{\Gamma;\Theta \vdash A + B \Leftarrow \mathtt{U}} \quad \text{(Sum)}$$

$$\frac{\Gamma;\Theta \vdash t_1 \Leftarrow \mathbf{A}}{\Gamma;\Theta \vdash \mathtt{inl}\ t_1 \Leftarrow \mathbf{A} + \mathbf{B}} \quad \text{(Sum-Inl)}$$

$$\frac{\Gamma;\Theta \vdash t_2 \Leftarrow \mathbf{B}}{\Gamma;\Theta \vdash \mathtt{inr}\ t_2 \Leftarrow \mathbf{A} + \mathbf{B}} \quad \text{(Sum-Inr)}$$

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

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

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

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

$$\frac{\Gamma \vdash e \Rightarrow [\psi']\mathbf{A}' \quad \psi \leq^\Theta \psi' \quad \mathbf{A} \sim^\Theta_{\tau(\Gamma)} \mathbf{A}'}{\Gamma \vdash e \Leftarrow [\psi]\mathbf{A}} \qquad \text{(Weak-Partial)}$$

$$\frac{\Gamma \vdash e \Rightarrow [\vec{\psi}' \mapsto \vec{\mathbf{t}}']\mathbf{A}' \quad [\vec{\psi} \mapsto \vec{\mathbf{t}}] \sim^{\Theta,\psi_1 \vee \cdots \vee \psi_n}_{\tau(\Gamma)} [\vec{\psi}' \mapsto \vec{\mathbf{t}}'] \quad \mathbf{A} \sim^\Theta_{\tau(\Gamma)} \mathbf{A}'}{\Gamma \vdash e \Leftarrow [\vec{\psi} \mapsto \vec{\mathbf{t}}]\mathbf{A}} \qquad \text{(Weak-Restr)}$$

$$\frac{\Gamma;\Theta \vdash e \Rightarrow \mathbf{A} \quad \mathbf{A} \sim^\Theta_{\tau(\Gamma)} \mathbf{A}'}{\Gamma;\Theta \vdash e \Leftarrow \mathbf{A}'} \qquad \text{(Infer)}$$

4.3 $\alpha\eta$-conversion

The function $\cdot \sim^\Theta_{ns} \cdot$ tests α and η 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.

The rules assume that the two values being tested have the same type.

$$\frac{}{\mathtt{U} \sim^\Theta_{ns} \mathtt{U}}$$

$$\frac{(ty)_\Gamma \sim^\Theta_{ns} (ty')_{\Gamma'} \quad (e)_{\Gamma, x \mapsto v^{(ty)_\Gamma}} \sim^\Theta_{ns,v} (e')_{\Gamma', y \mapsto v^{(ty')_{\Gamma'}}}}{\textsc{closure}([x : ty]e, \Gamma) \sim^\Theta_{ns} \textsc{closure}([y : ty']e', \Gamma')} \ (v = \textsc{newVar}(ns{+}{+}\Gamma', x))$$

$$\frac{(e)_{\Gamma, x \mapsto v^{(ty)_\Gamma}} \sim^\Theta_{ns,v} \textsc{app}(\textsc{simplify}(\mathbf{v}_2), v^{(ty)_\Gamma})}{\textsc{closure}([x : ty]e, \Gamma) \sim^\Theta_{ns} \mathbf{v}_2} \ (v = \textsc{newVar}(ns, x))$$

$$\frac{\textsc{app}(\textsc{simplify}(\mathbf{v}_1), v^{(ty)_\Gamma}) \sim^\Theta_{ns,v} (e)_{\Gamma, x \mapsto v^{(ty)_\Gamma}}}{\mathbf{v}_1 \sim^\Theta_{ns} \textsc{closure}([x : ty]e, \Gamma)} \ (v = \textsc{newVar}(ns, x))$$

$$\frac{(ty)_\Gamma \sim^\Theta_{ns} (ty')_{\Gamma'} \quad (e)_{\Gamma, x \mapsto v^{(ty)_\Gamma}} \sim^\Theta_{ns,v} (e')_{\Gamma', y \mapsto v^{(ty')_{\Gamma'}}}}{\textsc{closure}(\langle x : ty \rangle e, \Gamma) \sim^\Theta_{ns} \textsc{closure}(\langle y : ty' \rangle e', \Gamma')} \ (v = \textsc{newVar}(ns{+}{+}\Gamma', x))$$

$$\frac{\mathbf{s} \sim^{\Theta}_{ns} \mathbf{t}}{\mathbf{s}.1 \sim^{\Theta}_{ns} \mathbf{t}.1}$$

$$\frac{\mathbf{s} \sim^{\Theta}_{ns} \mathbf{t}}{\mathbf{s}.2 \sim^{\Theta}_{ns} \mathbf{t}.2}$$

$$\frac{\mathbf{s} \sim^{\Theta}_{ns} \mathbf{s}' \qquad \mathbf{t} \sim^{\Theta}_{ns} \mathbf{t}'}{(\mathbf{s}, \mathbf{t}) \sim^{\Theta}_{ns} (\mathbf{s}', \mathbf{t}')}$$

$$\frac{\textsc{fst}(\textsc{simplify}_{\Theta}(\mathbf{v})) \sim^{\Theta}_{ns} \mathbf{s} \qquad \textsc{snd}(\textsc{simplify}_{\Theta}(\mathbf{v})) \sim^{\Theta}_{ns} \mathbf{t}}{\mathbf{v} \sim^{\Theta}_{ns} (\mathbf{s}, \mathbf{t})}$$

$$\frac{\mathbf{s} \sim^{\Theta}_{ns} \textsc{fst}(\textsc{simplify}_{\Theta}(\mathbf{v})) \qquad \mathbf{t} \sim^{\Theta}_{ns} \textsc{snd}(\textsc{simplify}_{\Theta}(\mathbf{v}))}{(\mathbf{s}, \mathbf{t}) \sim^{\Theta}_{ns} \mathbf{v}}$$

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

$$\frac{}{\mathbb{I} \sim^{\Theta}_{ns} \mathbb{I}} \qquad \frac{}{\mathtt{0} \sim^{\Theta}_{ns} \mathtt{0}} \qquad \frac{}{\mathtt{1} \sim^{\Theta}_{ns} \mathtt{1}}$$

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

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

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

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

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

$$\frac{(x = y) \sim^{\Theta} \texttt{True}}{\mathbf{x}^{\mathbb{I}} \sim^{\Theta}_{ns} \mathbf{y}^{\mathbb{I}}} \qquad \frac{(x = 0) \sim^{\Theta} \texttt{True}}{\mathbf{x}^{\mathbb{I}} \sim^{\Theta}_{ns} \texttt{0}} \qquad \frac{(x = 1) \sim^{\Theta} \texttt{True}}{\mathbf{x}^{\mathbb{I}} \sim^{\Theta}_{ns} \texttt{1}} \qquad \frac{(x = 0) \sim^{\Theta} \texttt{True}}{\texttt{0} \sim^{\Theta}_{ns} \mathbf{x}^{\mathbb{I}}} \qquad \frac{(x = 1) \sim^{\Theta} \texttt{True}}{\texttt{1} \sim^{\Theta}_{ns} \mathbf{x}^{\mathbb{I}}}$$

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

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

$$\frac{\mathbf{F} \sim^{\Theta}_{ns} \mathbf{F}' \quad \varphi \sim^{\Theta} \varphi' \quad \mathbf{i}_0 \sim^{\Theta}_{ns} \mathbf{i}'_0 \quad \mathbf{u} \sim^{\Theta}_{ns} \mathbf{u}' \quad \mathbf{b} \sim^{\Theta}_{ns} \mathbf{b}' \quad \mathbf{i} \sim^{\Theta}_{ns} \mathbf{i}'}{\texttt{comp}\ \mathbf{F}\ \varphi\ \mathbf{i}_0\ \mathbf{u}\ \mathbf{b}\ \mathbf{i} \sim^{\Theta}_{ns} \texttt{comp}\ \mathbf{F}'\ \varphi'\ \mathbf{i}'_0\ \mathbf{u}'\ \mathbf{b}'\ \mathbf{i}'}$$

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

$$\frac{\psi_i \not\sim^{\Theta} \texttt{True} \quad \psi'_j \not\sim^{\Theta} \texttt{True} \quad \mathbf{k} \sim^{\Theta}_{ns} \mathbf{k}'}{\mathbf{k}^{[\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]\mathbf{A}} \sim^{\Theta}_{ns} \mathbf{k}'^{[\psi'_1 \mapsto \mathbf{t}'_1, \ldots, \psi'_m \mapsto \mathbf{t}'_m]\mathbf{A}'}} \begin{array}{l}(1 \leq i \leq n) \\ (1 \leq j \leq m)\end{array}$$

$$\frac{\textsc{proofIrrelevant}_{\Theta}(\mathbf{A})}{\mathbf{k}^{\mathbf{A}} \sim^{\Theta}_{ns} \mathbf{k}'^{\mathbf{A}'}}$$

Proof irrelevance

When two values have a type that can be simplified (i.e. a restriction type with a true formula), then both values shall reduce to the same value and so they are indeed $\alpha\eta$-equivalent.[7] We generalize this by introducing a predicate called PROOFIRRELEVANT: when we test $\alpha\eta$-equivalence between two neutral values, if their type is proof irrelevant it means that they shall reduce to the same value, and so they are automatically convertible (from this comes the name 'proof irrelevance', since we don't have to look inside the value/proof).

$$\frac{\psi_i \sim^{\Theta} \texttt{True}}{\textsc{proofIrrelevant}([\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]\mathbf{A})}$$

$$\frac{\textsc{proofIrrelevant}(\mathbf{A})}{\textsc{proofIrrelevant}([\psi_1 \mapsto \mathbf{t}_1, \ldots, \psi_n \mapsto \mathbf{t}_n]\mathbf{A})}$$

[7] This works because when we test $\alpha\eta$-equivalence, the two values being tested are assumed to have the same type.

$$\frac{\textsc{proofIrrelevant}((e)_{\Gamma,x\mapsto v})}{\textsc{proofIrrelevant}(\textsc{closure}([x:ty]e,\Gamma))}\ (v = \textsc{newVar}(\Gamma,x))$$

$$\frac{\textsc{proofIrrelevant}((ty)_{\Gamma})\quad \textsc{proofIrrelevant}((e)_{\Gamma,x\mapsto v})}{\textsc{proofIrrelevant}(\textsc{closure}(\langle x:ty\rangle e,\Gamma))}\ (v = \textsc{newVar}(\Gamma,x))$$