Frex^2: Normalisation by Evaluation for Second-Order Algebras

Greg Brown

February 16, 2023

1 Second-Order Algebra

Universal second-order algebra is a technique to describe syntax and theories for algebras with operations that can bind variables. One notable example of a second-order algebra is the simply-typed lambda calculus (STLC). Second-order algebras depend on a set T of types. For instance, the grammar $T \coloneqq N \mid T \rightarrowtail T$ gives the types for the STLC, where N describes some neutral base type (e.g. natural numbers).

A *signature* is a pair of a set O and function $|\cdot| \in O \to (T^* \times T)^* \times T$. The set O is the set of operators in the signature. The function gives the arity and return type of each operator. Each operator can take a number of arguments, hence the arity is a list of descriptors. Because arguments can bind variables, each descriptor has a list of bound variable types, as well as the argument type. Let the notation $(\gamma_i)\sigma_i \vdash o : \tau$ denote $|o| = (\gamma_i, \sigma_i)_i, \tau$.

The signature for the STLC has the two operators $\mathtt{app}_{\alpha,\beta}$ and $\mathtt{abs}_{\alpha,\beta}$ for each pair of types α and β. Their arities are $\alpha \rightarrowtail \beta, \alpha \vdash \mathtt{app}_{\alpha,\beta} : \beta$ and $(\alpha)\beta \vdash \mathtt{abs}_{\alpha,\beta} : \alpha \rightarrowtail \beta$. Application (`app`) takes two arguments—a function and value—and returns a result. Neither argument to application binds variables. Abstraction (`abs`) takes a single argument, whose type is the output type of the function. This argument has access to a freshly-bound variable of the input type.

The unpairing operation in STLC with products is also a second-order operation. The operation `let (x, y) = e in e'` is represented by an operator $\mathtt{unpair}_{\alpha,\beta,\tau}$, with $\alpha \times \beta, (\alpha, \beta)\tau \vdash \mathtt{unpair}_{\alpha,\beta,\tau} : \tau$. The argument `e` is a pair and has no additional variables in scope. The second argument `e'` has access to two new variables, corresponding to the two components of the pair `x` and `y`[1].

[1] Think of a good non-example.

Define a *sorted family* to be a type-and-context-indexed set. For example, the family of variables $\mathcal{I}\,\tau\,\Gamma$ is the set of positions of type τ in the context Γ. Write $\Gamma \vdash^{\mathcal{X}} \mathtt{t} : \tau$ for $\mathtt{t} \in \mathcal{X}\,\tau\,\Gamma$, for any sorted family $\mathcal{X}$.

Given a signature $\Sigma = (O, |\cdot|)$, an *algebra* has four constituent parts:

- a carrier sorted family $\mathcal{A}$
- an operation $[\![\cdot]\!]_o \in (\mathcal{A}\,\sigma_i\,(\gamma_i + \Gamma))_i \to \mathcal{A}\,\tau\,\Gamma$ for each operator $(\gamma_i)\sigma_i \vdash o : \tau$
- a mapping $\mathtt{var} \in \mathcal{I}\,\tau\,\Gamma \to \mathcal{A}\,\tau\,\Gamma$
- a mapping $\mathtt{sub} \in \mathcal{A}\,\tau\,\Gamma \to (\forall\sigma.\mathcal{I}\,\sigma\,\Gamma \to \mathcal{A}\,\sigma\,\Delta) \to \mathcal{A}\,\tau\,\Delta$

subject to the following conditions:

left unit: $\mathtt{sub}(\mathtt{var}_i, \sigma) = \sigma_i$

right unit: $\mathtt{sub}(\mathtt{t}, \mathtt{var}) = \mathtt{t}$

associativity: $\mathtt{sub}(\mathtt{sub}(\mathtt{t}, \sigma), \varsigma) = \mathtt{sub}(\mathtt{t}, i \mapsto \mathtt{sub}(\sigma_i, \varsigma))$

naturality: $\mathtt{sub}([\![\mathtt{ts}]\!]_o, \sigma) = [\![(\mathtt{sub}(\mathtt{ts}_i, \Uparrow \sigma))_i]\!]_o$

where $\Uparrow \sigma$ uses $\mathtt{var}$ and $\mathtt{sub}$ to lift a substitution σ from $\mathcal{I}\,\tau\,\Gamma \to \mathcal{A}\,\tau\,\Delta$ to $\mathcal{I}, \tau\,(\Theta + \Gamma) \to \mathcal{A}, \tau\,(\Theta + \Delta)$. The dual extension, $\sigma \Uparrow \in \mathcal{I}, \tau\,(\Gamma + \Theta) \to \mathcal{A}, \tau\,(\Gamma + \Theta)$ will be used later.

The associativity, left and right unit conditions assert that $\mathcal{A}$ is a *substitution monoid*. This means that substitution interacts with itself and variables in an intuitive way. The naturality condition asserts that substitution passes through operators and is capture-avoiding. Substitution lifting maps freshly-bound variables to themselves.

For the STLC, an algebra $\mathcal{A}$ has a carrier $\mathcal{A}$; a pair of operations $[\![\cdot]\!]_{\mathsf{app}} \in \mathcal{A}\,(\alpha \rightarrowtail \beta)\,\Gamma \to \mathcal{A}\,\alpha\,\Gamma \to \mathcal{A}\,\beta\,\Gamma$, and $[\![\cdot]\!]_{\mathsf{abs}} \in \mathcal{A}\,\beta\,(\alpha, \Gamma) \to \mathcal{A}\,(\alpha \rightarrowtail \beta)\,\Gamma$; and the variable and substitution maps.

One example algebra for any signature is the unit algebra $\mathbf{1}$, where $\mathbf{1}\,\tau\,\Gamma = \{*\}$. Each operation $[\![\cdot]\!]_o$, and the maps $\mathtt{var}$ and $\mathtt{sub}$, return the unique value of the appropriate type. This trivially satisfies the conditions.

Another example is the algebra of sets and functions for the STLC. First, take any set V to represent values of type N. A type T can then be interpreted as a set of values $\mathcal{V}[T]$:

$$\mathcal{V}[N] \mapsto V$$
$$\mathcal{V}[\alpha \rightarrowtail \beta] \mapsto \mathcal{V}[\alpha] \to \mathcal{V}[\beta]$$

We can take the carrier sorted family $\mathcal{A}\,\tau\,\Gamma = (\forall\sigma.\mathcal{I}\,\sigma\,\Gamma \to \mathcal{V}[\sigma]) \to \mathcal{V}[\tau]$. From here, defining the operations and mappings is straight forward:

$$\begin{aligned}
[\![f, g]\!]_{\mathsf{app}}(\gamma) &= f(\gamma, (g(\gamma))) \\
[\![f]\!]_{\mathsf{abs}}(\gamma) &= x \mapsto f(\texttt{here} \mapsto x; \texttt{there}\, i \mapsto \gamma\, i) \\
\texttt{var}(i, \gamma) &= \gamma_i \\
\texttt{sub}(f, \sigma, \gamma) &= f\,(i \mapsto \sigma_i(\gamma))
\end{aligned}$$

Verifying that this definition satisfies the coherence conditions is left as an exercise[2].

2 Metavariables

Here is a typical way of writing the β-reduction relation for STLC:

```
(λ x. t) $ u ⤳ t[x ↦ u]
```

The variable `t` is implicitly parameterised by the bound variable `x`, meaning $\alpha, t \vdash \texttt{t} : \beta$. In contrast, `u` is not parameterised by any terms. We can make the parameters explicit:

```
(λ x. t<x>) $ u<> ⤳ t<u<>>
```

`t` and `u` are *metavariables*—free variables with parameters. We can instantiate metavariables with concrete terms to create new ones. This property is formalised in the definition of a *syntactic algebra*.

A syntactic algebra for a signature Σ over a sorted family of metavariables $\mathfrak{X}$ is:

- an algebra $\mathcal{A}$ over the signature
- a map $\texttt{mvar} \in \mathfrak{X}\,\tau\,\Pi \to (\forall\sigma.\mathcal{I}\,\sigma\,\Pi \to \mathcal{A}\,\sigma\,\Gamma) \to \mathcal{A}\,\tau\,\Gamma$
- and a map $\texttt{msub} \in \mathcal{A}\,\tau\,\Gamma \to (\forall\sigma,\Theta.\mathfrak{X}\,\sigma\,\Theta \to \mathcal{A}\,\sigma\,(\Theta + \Delta)) \to \mathcal{A}\,\tau\,(\Gamma + \Delta)$

which satisfy these conditions[3]:

1. $\texttt{sub}(\texttt{mvar}(\mathfrak{m}, \sigma), \varsigma) = \texttt{mvar}(\mathfrak{m}, i \mapsto \texttt{sub}(\sigma_i, \varsigma))$
2. $\texttt{msub}([\![ts]\!]_o, \zeta) = [\![\texttt{msub}(ts_i, \zeta)]\!]_o$
3. $\texttt{msub}(\texttt{var}(i), \zeta) = \texttt{var}(\texttt{inl}(i))$
4. $\texttt{msub}(\texttt{sub}(t, \sigma), \zeta) = \texttt{sub}(\texttt{msub}(t, \zeta), \sigma \Uparrow)$
5. $\texttt{sub}(\texttt{msub}(t, \zeta), \Uparrow \sigma) = \texttt{msub}(t, \mathfrak{m} \mapsto \texttt{sub}(\zeta_{\mathfrak{m}}, \Uparrow \sigma))$

[2]This is me being lazy.

[3]Explain why these conditions exist, and why they are so ugly.

6. $\texttt{msub}(\texttt{mvar}(\mathfrak{m}, \sigma), \zeta) = \texttt{sub}(\zeta_{\mathfrak{m}}, (i \mapsto \texttt{msub}(\sigma_i, \zeta)) + \texttt{inr} \circ \texttt{var})$
7. $\texttt{msub}(\texttt{msub}(t, \zeta), \xi) = \texttt{msub}(t, \mathfrak{m} \mapsto \texttt{msub}(\zeta_{\mathfrak{m}}, \xi))$

The map `mvar` takes a metavariable $\Pi \vdash^{\mathfrak{X}} \mathfrak{m} : \tau$ and converts it into a term in context Γ. This is by giving a value for each of $\mathfrak{m}$'s parameters, achieved via the substitution argument.

The syntactic substitution map `msub` is more complex. First note that it is linear: the output context is an extension of the input. This means that metasubstitution can use bound variables, as long as they were bound in a higher scope. Also observe that the substitution map (the second parameter) is independant of the input context Γ. Because metavariables are always associated with a substitution, it is sufficient to map only the metavariables in their original contexts. The context can be "corrected" by a substitution.

The term algebra $\mathbb{T}[\mathfrak{X}]$ for a signature is the free syntactic algebra over $\mathfrak{X}$, with for each map $f \in \mathfrak{X}\,\tau\,\Gamma \to \mathcal{A}\,\tau\,\Gamma$ a unique homomorphism $\texttt{bind}(f) \in \mathbb{T}[\mathfrak{X}] \to \mathcal{A}$.

Return to the earlier example of β reduction. We have metavariables $\mathfrak{X}$ with two elements: $\alpha \vdash^{\mathfrak{X}} \texttt{t} : \beta$ and $\vdash^{\mathfrak{X}} \texttt{u} : \alpha$. We then have two terms, both of type β in an empty context: `(λ x. t<x>) $ u<>` and `t<u<>>`.

As a more complex example, we can construct a syntactic substitution map $\zeta \in \mathfrak{X}\,\sigma\,\Theta \to \mathcal{A}\,\sigma\,(\Theta + \alpha \rightarrowtail \beta, \alpha)$[4]:

$$\alpha, \alpha \rightarrowtail \beta, \alpha \vdash^{\mathcal{A}} \texttt{x}_1 \;\texttt{\$}\; \texttt{x}_0 : \beta$$
$$\alpha \rightarrowtail \beta, \alpha \vdash^{\mathcal{A}} \texttt{x}_1 : \alpha$$

Metavariables allow a formal definition of equational theories, and then models. An *equation* is a quintuple $(\mathfrak{X}, \Gamma, \tau, \texttt{t}, \texttt{u})$ such that $\Gamma \vdash^{\mathbb{T}[\mathfrak{X}]} \texttt{t} : \tau$ and $\Gamma \vdash^{\mathbb{T}[\mathfrak{X}]} \texttt{u} : \tau$. An *equational theory* is a signature with a finite set of axioms $\Gamma \vdash^{\mathbb{T}[\mathfrak{X}]} \texttt{t} =_a \texttt{u} : \tau$. This can be extended to an equivalence relation $\Gamma \vdash^{\mathbb{T}[\mathfrak{X}]} \texttt{t} = \texttt{u} : \tau$ by taking the closure of axioms with respect to metasubstitution.

A *model* of an equational theory is an algebra $\mathcal{A}$ where all equations are valid: given an equation $\Gamma \vdash^{\mathbb{T}[\mathfrak{X}]} \texttt{t} = \texttt{u} : \tau$, then for any environment map $f \in \mathfrak{X}\,\tau\,\Gamma \to \mathcal{A}\,\tau\,\Gamma$, the equality $\texttt{bind}(f, t) = \texttt{bind}(f, u)$ holds.

3 Free Extensions

Given an algebra (or model) $\mathcal{A}$ and metavariables $\mathfrak{X}$, a *free extension* of $\mathcal{A}$ by $\mathfrak{X}$ is a syntactic algebra (or model) $\mathcal{F}$ over $\mathfrak{X}$ with the following additional structure:

[4]Include some examples.

- a homomorphism $\texttt{sta} \in \mathcal{A} \to \mathcal{F}$
- for every pair of homomorphism $f \in \mathcal{A} \to \mathcal{B}$ and map $g \in \mathcal{X}\,\tau\,\Gamma \to \mathcal{B}\,\tau\,\Gamma$, a unique homomorphism $\texttt{interp}(f, g) \in \mathcal{F} \to \mathcal{B}$

with the following two conditions:

- $\texttt{interp}(f, g) \circ \texttt{sta} = g$
- $\texttt{interp}(f, g, \texttt{mvar}(\mathfrak{m}, \sigma)) = \texttt{sub}(g(\mathfrak{m}), \texttt{interp}(f, g) \circ \sigma)$

This is equivalently the algebraic (model) coproduct of $\mathcal{A}$ and $\mathbb{T}[\mathfrak{X}]$. The free extension is an extension in the sense that the algebra $\mathcal{A}$ is given an interpretation of metavariables. It is free in the sense that adding the metavariables minimally restricts the homomorphisms to or from the algebra.

Define the map $\texttt{dyn} \in \mathfrak{X}\,\tau\,\Gamma \to \mathcal{F}\,\tau\,\Gamma$ by $\texttt{dyn}(\mathfrak{m}) = \texttt{msub}(\mathfrak{m}, \texttt{var})$. The names `sta` and `dyn` come from the perspective of staged compilation: values made with `sta` are known statically and can be acted on during compilation. Values defined via `dyn` are only known at runtime and have no inherent algebraic structure.

Using `sta` and `dyn`, there is a homomorphism $\texttt{eval} \in \mathbb{T}[\mathcal{A} \cup \mathfrak{X}] \to \mathcal{F}$ given by $\texttt{eval} = \texttt{bind}(\texttt{sta} \cup \texttt{dyn})$. Because there is a morphism $\mathcal{A} \to \mathbb{T}[\mathcal{A}]_{/\approx}$[5], there is also a morphism $\texttt{reify} \in \mathcal{F} \to \mathbb{T}[\mathcal{A} \cup \mathfrak{X}]_{/\approx}$ defined using `interp`.

For brevity, let $\overline{\texttt{t}}$ represent $\texttt{sta}(\texttt{t})$ and understand $\underline{\mathfrak{m}}$ to be $\texttt{dyn}(\mathfrak{m})$.

4 βη-Normal Form as a Free Extension

In the STLC, βη-normal form is a way of writing terms such that test for βη equivalence is a simple identity check. I will adapt this normal form to attempt to produce a free extension of a STLC model $\mathcal{A}$ by metavariables $\mathfrak{X}$, dubbed $\mathcal{A}_{\beta\eta}[\mathfrak{X}]$.

A term is *dynamic* if there is a subterm that uses a dynamic metavariable. Otherwise it is static. For example, $\underline{\mathfrak{m}}\langle\zeta\rangle$ and $\overline{\texttt{t}}\ \texttt{\$}\ (\lambda\texttt{x}.\ \underline{\texttt{n}}\langle\texttt{x}\rangle)$ are both dynamic, whilst $\texttt{x}_i$, $\overline{\texttt{t}}$ and $\lambda\texttt{x}.\ \overline{\texttt{u}}\langle\texttt{x}\rangle$ are static.

The sorted family $\mathcal{A}_{\beta\eta}[\mathfrak{X}]$ and a family $\mathcal{A}_{ne}[\mathfrak{X}]$ are defined inductively by the following judgements:

[5]Define what $\approx$ is.

$$\dfrac{\mathtt{x} : \alpha, \Gamma \vdash^{\mathcal{A}_{\beta\eta}[\mathfrak{X}]} \mathtt{t} : \beta}{\Gamma \vdash^{\mathcal{A}_{\beta\eta}[\mathfrak{X}]} \lambda\mathtt{x}.\ \mathtt{t} : \alpha \rightarrowtail \beta}\ 1 \qquad \dfrac{\Gamma \vdash^{\mathcal{A}_{ne}[\mathfrak{X}]} \mathtt{t} : N}{\Gamma \vdash^{\mathcal{A}_{\beta\eta}[\mathfrak{X}]} \mathtt{t} : N}\ 2$$

$$\dfrac{\Gamma \vdash^{\mathcal{A}_{ne}[\mathfrak{X}]} \mathtt{t} : \alpha \rightarrowtail \beta \qquad \Gamma \vdash^{\mathcal{A}_{\beta\eta}[\mathfrak{X}]} \mathtt{u} : \alpha \qquad \mathtt{t} \text{ or } \mathtt{u} \text{ uses a dynamic variable}}{\Gamma \vdash^{\mathcal{A}_{ne}[\mathfrak{X}]} \mathtt{t}\ \$\ \mathtt{u} : \beta}\ 3$$

$$\dfrac{\Pi \vdash^{\mathfrak{X}} \mathfrak{m} : \tau \qquad \Gamma \vdash^{\mathcal{A}_{ne}[\mathfrak{X}]} \sigma_i : \Pi_i}{\Gamma \vdash^{\mathcal{A}_{ne}[\mathfrak{X}]} \underline{\mathfrak{m}}\langle\sigma\rangle : \tau}\ 4 \qquad \dfrac{\Gamma \vdash^{\mathcal{A}} \mathtt{t} : \tau}{\Gamma \vdash^{\mathcal{A}_{ne}[\mathfrak{X}]} \overline{\mathtt{t}}\langle\mathtt{var}\rangle : \tau}\ 5$$

$$\dfrac{\Pi \vdash^{\mathcal{A}} \mathtt{t} : \tau \qquad \Gamma \vdash^{\mathcal{A}_{\beta\eta}[\mathfrak{X}]} \sigma_i : \Pi_i \qquad \sigma \text{ uses a dynamic variable}}{\Gamma \vdash^{\mathcal{A}_{ne}[\mathfrak{X}]} \overline{\mathtt{t}}\langle\sigma\rangle : \tau}\ 6$$

In summary, this normal form first requires terms to have no β reductions (resulting in a family of neutral terms $\mathcal{A}_{ne}[\mathfrak{X}]$). Then terms are η expanded so each type constructor has exactly one constructor.

Rules 1 and 2 describe how to convert neutral terms into normal forms: each function requires an explicit abstraction, and terms of neutral type must be neutral.

Rule 3 describes which applications are irreducible. The first argument to the application must itself be irreducible. If the only requirement was for it to be in normal form, then it could be an abstraction term. This is β-reducible, so goes against the principal of performing all possible reductions. Further at least one argument must use a dynamic variable. If both arguments to application were static, then they are both values in $\mathcal{A}$ so the application can be reduced by using application from $\mathcal{A}$. Thus, evaluation is stuck only if at least one of the two argument has a dynamic variable.

Rule 4 states that dynamic variables are always stuck. There is no information to allow for further evaluation, so they are always neutral.

Rules 5 and 6 describe which substitutions into static values are neutral. Rule 5 says that the identity substitution is stuck. Static values always appear with a substitution in terms. Because applying the identity substitution doesn't change the static value, nor does it influence any later substitutions, it is the maximally-reduced substitution where all components are static. The hypothesis of rule 6 requires at least one substitution parameter to be dynamic. Vecause substitutions are performed in parallel, this blocks the full substitution.

Notice that neither normal forms or neutral terms used bound variables. Any variable $\mathtt{x}_i$ is equivalent to $\overline{\mathtt{x}_i}\langle\mathtt{var}\rangle$. Expanding bound variables into static values can be seen as a reduction, so should be evaluated according to the principal of performing all possible reductions.

Using a logical relations argument, it is possible to define a mapping $\mathbb{T}[\mathcal{A} \cup \mathfrak{X}] \to \mathcal{A}_{\beta\eta}[\mathfrak{X}]$[6].

5 The Equality Problem

Unfortunately, the normal form just presented using equality is not a model of STLC. In fact, it is not even an algebra. The problem comes from substitution, and can be demonstrated by the following two equal terms:

$$\begin{aligned} \overline{\mathtt{x}_i \ \$ \ \mathtt{t}}\langle\underline{\sigma}\rangle &= (\overline{\mathtt{x}_i} \ \$ \ \overline{\mathtt{t}})[\underline{\sigma}] \\ &= (\mathtt{x}_i \ \$ \ \overline{\mathtt{t}})[\underline{\sigma}] \\ &= \mathtt{x}_i[\underline{\sigma}] \ \ \$ \ \overline{\mathtt{t}}[\underline{\sigma}] \\ &= \underline{\sigma_i} \ \$ \ \overline{\mathtt{t}}\langle\underline{\sigma}\rangle \end{aligned}$$

Assuming the terms have type N, then they are in normal form. The top two lines normalise to the first term, and the bottom two lines normalise to the final term. However, because the subterm $\overline{\mathtt{x}_i \ \$ \ \mathtt{t}}$ is opaque for an arbitrary algebra $\mathcal{A}$, these two normal forms for the same term are distinct.

For this normal form to be an algebra, it must be taken quotient $\approx$, the same relation the term algebra is subject to.

6 Removing the Quotient: Effective Free Extensions

In practical systems, using a quotient for equality can be undesirable. Especially when the quotient is undecidable. An *effective* free extension uses identity of values for equality—both easy to compute and decidable.

I suspect that there is no free extension for STLC that is effective for all algebras, due to the equality problem. Instead, I will discuss classes of algebras where the βη-normal form is effective.

The simplest case is when $\mathcal{A} = \mathbb{T}[\varnothing]$: the initial algebra. In this case, $\mathbb{T}[\mathbb{T}[\varnothing] \cup \mathfrak{X}]_{/\approx} \cong \mathbb{T}[\mathfrak{X}]$, and $\mathbb{T}[\varnothing]_{\beta\eta}[\mathfrak{X}]$ is the classical βη-normal form extended with metavariables. By design, βη-normal form identifies terms that are equivalent, thus this is an effective free extension. This argument can be generalised to $\mathcal{A} = \mathbb{T}[\mathfrak{Y}]$, for any fixed set of metavariables $\mathfrak{Y}$, as $\mathbb{T}[\mathbb{T}[\mathfrak{Y}] \cup \mathfrak{X}]_{/\approx} \cong \mathbb{T}[\mathfrak{Y} \cup \mathfrak{X}]$.

Now consider an algebra $\mathcal{A}$ such that there is an isomorphism $\mathcal{A} \cong \mathbb{T}[\mathfrak{Y}]$. This is true for all "normal forms" of $\mathbb{T}[\mathfrak{Y}]$, such as weak-head normal form,

[6]How, exactly?

$\beta\eta$-normal form, and CPS form. $\mathcal{A}_{\beta\eta}[\mathfrak{X}]$ is an effective free extension, because it is isomorphic to $\mathbb{T}[\mathfrak{Y}]_{\beta\eta}[\mathfrak{X}]$, which is effective.

One thing missing from the previous form is terms that compute on values of the neutral type. For example, if an algebra $\mathcal{A}$ treats values of type N as naturals, with an addition operator, then $\mathcal{A} \not\rightarrow \mathbb{T}[\mathbb{N} \cup \{+\}]$, because $\overline{5} = \overline{add\langle 2, 3\rangle} = \overline{add}\langle \overline{2}, \overline{3}\rangle$, but the outer terms are not equal in $\mathbb{T}[\mathbb{N} \cup \{+\}]$.

Everthing past here is a rough sketch.

Define $\mathcal{A}$ to be a (sub|super)-model[7] of the STLC when there is:

- a sorted family $\mathfrak{Y}$
- a set of axioms $\mathbf{Ax} \in \varnothing \vdash^{\mathbb{T}[\mathfrak{X} \cup \mathfrak{Y}]} t =_{\mathcal{A}} u : \tau$
- an isomorphism $\mathcal{A} \cong \mathbb{T}[\mathfrak{Y}]_{/\mathbf{Ax}}$

where

- abstraction does not occur in $\mathbf{Ax}$
- all equalities in $\mathcal{A}$ are generated by β- and η- equalities and $\mathbf{Ax}$

A Commentary about LaTeX

The character $\mathfrak{Y}$ is written `\mathfrak{Y}`, yet it looks much more like an “N” to me.

[7] I don’t know which is better. Maybe it should be something completely different?