Unification with binding

May 4, 2022

We show correctness of an algorithm computing the most general unifier of a list of term pairs involving binding operations and n-ary metavariables applied to distinct variables. We reason in the framework of Σ-monoids [3].

1 Notations

In a category with coproducts, we denote by $in : A_i \hookrightarrow \coprod_i A_i$ the coproduct injection.

Given a monad T on a category C, we denote the Kleisli category by Kl_T: objects are objects of C, and morphisms between c and c' are morphisms in C between c and Tc'. Note that there is a bijection between Kleisli morphisms $c \to Tc'$ and T-algebra morphisms $Tc \to Tc'$. We denote by f^* the T-algebra morphism induced by a Kleisli morphism. Composition of g and f is given by $g^* \circ f$.

2 Setting

Let Σ be a binding signature. Then, there is a free Σ-monoid monad T on $Set^{\mathbb{N}}$ such that $T(X)$ is the syntax of Σ extended with an n-ary operation for each $x \in X_n$. More precisely, $T(X)_n$ is the set of terms whose free variables are in $\{0, \ldots, n-1\}$.

Example 1. Consider the case of λ-calculus. Then, $T(\emptyset)_n$ is the set of λ-terms t taking free variables in $\{0, \ldots, n-1\}$.

Consider $X \in Set^{\mathbb{N}}$ such that $X_2 = \{M\}$ and $X_{n\neq 2} = \emptyset$. Then $T(X)_n$ is the set of λ-terms t involving a metavariable M of arity 2 such that $fv(t) \subset \{0, \ldots, n-1\}$.

Consider $Y \in Set^{\mathbb{N}}$ such that $Y_2 = \{M, N\}$, $Y_0 = \{C\}$ and $Y_n = \emptyset$ otherwise. Then $T(Y)_n$ is the λ-term t with metavariables M,N of arity 2 and a constant metavariable C such that $fv(t) \subset \{0, \ldots, n-1\}$.

Note that metavariables are allowed to be applied to arbitrary terms here, not only (distinct) variables.

Definition 2. A morphism $X \to T(Y)$ is said *safe* if it factors through $T'(Y) \to T(Y)$, where $T'(Y)_n$ is the subset of $T(Y)_n$ consisting of terms that where metavariables are applied to distinct variables only, rather than arbitrary terms.

Notation 3. We denote by Kl_T the Kleisli category of the monad T: objects are families in $Set^{\mathbb{N}}$ and a morphism $X \to Y$ is a family morphism $X \to T(Y)$.

Remark 4. A (metavariable) substitution is precisely a morphism in the Kleisli category. For example, a Kleisli morphism from X to Y is a morphism $X \to T(Y)$, i.e., for each $n \in \mathbb{N}$, a map $X_n \to T(Y)_n$ which assigns a term taking free variables in $\{0, \dots, n-1\}$ for each n-ary metavariable.

Definition 5. A family $X \in Set^{\mathbb{N}}$ is said *finite* if $\coprod_n X_n$ is finite.

Our main result consists in the following.

Theorem 6. *Let $V \rightrightarrows T(W)$ be a pair of safe morphisms between finite families V and W. If there is a coequalising Kleisli morphism, then there is coequaliser in $Kl(T)$.*

We use an explicit construction, detailed in the next section. This theorem allows to compute the most general unifier of two terms.

Corollary 7. *Let $t, u \in T(V)_n$ for some finite family $V \in Set^{\mathbb{N}}$ (thought as a specification of metavariables). If there exists a substitution unifying t and u, then there exists a most general unifier, i.e., a substitution $V \to T(W)$ such that any other unifying substitution uniquely factors through it.*

Proof. Giving two terms $t, u \in T(V)_n$ amounts to giving two parallel morphisms $yn \rightrightarrows T(V)$, where yn denotes the family defined by $yn_p = \emptyset$ if $p \neq n$ and yn_n is a singleton set. The most general unifier, if it exists, is the coequaliser of t and u. □

More generally, we can compute any finite colimit under the same condition.

Corollary 8. *Let $J : D \to Kl_T$ be a finite diagram selecting safe Kleisli morphisms and finite families. If there is a cocone, then J has a colimit.*

Proof. This follows from the computation of colimits as coequalisers and coproducts [4, Theorem V.2.2]. □

Let us finish this section by introducing some useful definitions, notations, and lemmas.

Lemma 9. *Free T-algebras extend to functors $\mathbb{F} \to Set$ preserving pullbacks, where $\mathbb{F}$ is the full subcategory of sets consisting in finite cardinals. Action on morphisms is given by renaming. Moreover, Kleisli morphisms are compatible with renaming.*

Proof. See Appendix A. □

Notation 10. If $n \in \mathbb{N}$, we denote the n^{th} cardinal set $\{0, \dots, n-1\}$ by n.

If $n \in \mathbb{N}$, we denote by yn the yoneda embedding into $Set^{\mathbb{N}}$, i.e., $yn(p)$ is empty if $n \neq p$ and a singleton set otherwise. We call *representable* any family which is isomorphic to some yn.

$I \in Set^{\mathbb{N}}$ denotes the family $I_n = n$.

Lemma 11. *Any family $X \in Set^{\mathbb{N}}$ is isomorphic to a coproduct of representable families. A family X is finite if and only if such a coproduct is finite.*

Proof. Clearly, any family X is isomorphic to $\coprod_n X_n yn \simeq \coprod_n \coprod_{x \in X_n} yn$. □

Notation 12. We represent a finite family $X \in Set^{\mathbb{N}}$ as a finite (metavariable) context $x_1 : n_1, \dots, x_p : n_p$ where $x_i \in X_{n_i}$.

3 Algorithm

The algorithm takes as input

- a (finite) metavariable context $\Gamma \in Set^{\mathbb{N}}$, that we denote by a list $M_1 : m_1, \dots M_n : m_n$ of metavariable symbols M_i with their associated arities m_i;
- a list of term pairs $t_i =_{n_i} u_i$, where n_i that t_i and u_i takes free variables in $\{0, \dots, n_i - 1\}$

It outputs:

- a metavariable context Δ
- a Kleisli map $\sigma : \Gamma \to T(\Delta)$, that is a substitution assigning to each metavariable $M : m$ declared in Γ a term with m free variables involving metavariables Δ.

In this section we inductively specify the algorithm through the judgement

$$\Gamma \vdash t_1 =_{n_1} u_1, \dots t_p =_{n_p} u_p \Rightarrow \sigma \dashv \Delta$$

that we sometimes abbreviate as

$$\Gamma \vdash \vec{t} =_{\vec{n}} \vec{u} \Rightarrow \sigma \dashv \Delta$$

The completeness statement (proven by induction) is the following.

Proposition 13. *Given a list of safe term pairs $\vec{t} = \vec{u}$ taking metavariables in Γ, if there exists a (finite) derivation tree of $\Gamma \vdash \vec{t} =_{\vec{n}} \vec{u} \Rightarrow \sigma \dashv \Delta$, then σ is the most general unifier. Moreover such a derivation tree exists as long as there exists at least one unifier.*

We will justify this proposition by showing that each introduced rule is sound, in the sense that that it supports the induction step.

Note that at most one rule is applyable given an input $\vec{t} = \vec{u}$.

Proposition 14. *There is at most one derivation tree of* $\Gamma \vdash \vec{t} =_{\vec{n}} \vec{u} \Rightarrow \sigma \dashv \Delta$.

Let us list indeed the rules according to the input $\vec{t} = \vec{u}$ in the conclusion:

- Empty list $\vec{t} = \vec{u} = ()$, in Section 3.1;
- Non empty, non singleton lists, $t_0 = u_0, \vec{t} = \vec{u}$, in Section 3.2;
- $M(\vec{x}) = N(\vec{y})$, in Section 3.3;
- $o(\vec{t}) = o(\vec{u})$, in Section 3.4;
- $M(\vec{x}) = o(\vec{t})$, in Section 3.5;
- $M(\vec{x}) = x_i$, in Section 3.6.

Let us mention the missing cases, which can never be unified anyway.

- $o(\vec{t}) = o'(\vec{u})$ when $o \neq o'$;
- $M(\vec{x}) = y$ when $y \notin \vec{x}$

From this list it follows

3.1 Empty list

$$\frac{}{\Gamma \vdash () \Rightarrow id \dashv \Gamma}$$

Soundness is straightforward.

3.2 Stepwise construction

We can restrict to the case where of a single coequaliser.

$$\frac{\Gamma \vdash t_0 =_{n_0} u_0 \Rightarrow \sigma_0 \dashv \Delta_1 \qquad \Delta_1 \vdash \vec{t}[\sigma_0] =_{\vec{n}} \vec{u}[\sigma_0] \Rightarrow \sigma \dashv \Delta_2 \qquad \vec{t} \text{ is not empty}}{\Gamma \vdash t_0 =_{n_0} u_0, \vec{t} =_{\vec{n}} \vec{u} \Rightarrow \sigma \circ \sigma_0 \dashv \Delta_2}$$

Soundness of this rule follows from the following general categorical lemma.

Lemma 15. *Let* $f_1, g_1 : A_1 \to B$ *and* $f_2, g_2 : A_2 \to B$ *be morphisms in some category. Then the coequaliser of the induced parallel morphism* $A_1 \coprod A_2 \rightrightarrows B$ *is the morphism* $B \to C \to D$ *defined as follows (assuming the involved coequalisers exist):*

1. $B \to C$ *is the coequaliser of* $A_1 \rightrightarrows B$*;*
2. $C \to D$ *is the coequaliser of* $A_2 \rightrightarrows B \to C$*.*

Corollary 16. *The above rule is sound.*

Proof. Assume there is a common unifier for $t_0 = u_0, \vec{t} = \vec{u}$. By induction hypothesis, the premises are derivable and thus the rule can be applied. Moreover, again by induction hypothesis, the premises produce most general unifiers. The output substitution of the conclusion is also the most general unifier, thanks to the previous lemma by specialising it to the Kleisli category Kl_T, taking $A_1 = yn_0$ and $A_2 = \coprod_i yn_i$. □

3.3 Case $M(x_1, \ldots, x_m) =_q N(y_1, \ldots, y_n)$

We need to make the distinction whether $M = N$ or not.

$$\frac{(i_1, \ldots, i_p) \text{ is the family of common positions: } x_{\vec{i}} = y_{\vec{i}}}{\Gamma \vdash M(\vec{x}) =_q M(\vec{y}) \Rightarrow M(1, \ldots, m) \mapsto N(i_1, \ldots i_p) \dashv \Gamma \backslash \{M\}, N : p}$$

The premise states that $i : p \to m$ is the equaliser of $\vec{x}, \vec{y} : m \to q$.

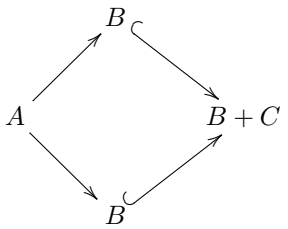
$$\frac{M \neq N \qquad (i_1, \ldots, i_p) \text{ and } (j_1, \ldots, j_p) \text{ are maximal such that } x_{\vec{i}} = y_{\vec{j}}}{\Gamma \vdash M(\vec{x}) =_q N(\vec{y}) \Rightarrow M(1, \ldots, m) \mapsto O(i_1, \ldots i_p), N(1, \ldots, n) \mapsto O(j_1, \ldots, j_p) \dashv \Gamma \backslash \{M, N\}, O : p}$$

The premise states that $m \xleftarrow{i} p \xrightarrow{j} n$ is a pullback of $m \xrightarrow{\vec{x}} q \xleftarrow{\vec{y}} n$.

Proposition 17. *The above rules are sound.*

Proof. We prove that the output substitution is the most general unifier.

Let us consider the first rule. We decompose Γ as $M : m, \Gamma'$ the coproduct of ym and Γ'. The term $M(\vec{x})$ corresponds to the composition of $yp \xrightarrow{M(\vec{x})} T(ym) \hookrightarrow T(\Gamma)$. More abstractly, we want to compute the coequaliser, in Kl_T, of



for $A = yp$, $B = ym$, $C = \Gamma'$. It is enough to compute the coequaliser of $f : B \to D$ of $A \rightrightarrows B$, for the desired coequaliser is then $B + C \xrightarrow{f+C} D + C$. So we need to compute the coequaliser of $yp \rightrightarrows T(ym)$. Thanks to Lemma 27, the coequaliser is $ym \xrightarrow{\mathbf{y}i} T(yp)$, where $i : p \to m$ is the equaliser of $\vec{x}$ and $\vec{y}$.

The second rule can be justified similarly. □

3.4 Case $o(\vec{t}) = o(\vec{u})$

$$\frac{o \text{ has binding arity } \vec{n} \qquad \Gamma \vdash \vec{t} =_{q+\vec{n}} \vec{u} \Rightarrow \sigma \dashv \Delta}{\Gamma \vdash o(\vec{t}) =_q o(\vec{u}) \Rightarrow \sigma \dashv \Delta}$$

Lemma 18. *This rule is sound.*

Proof. The result follows easily from the fact that a substitution unifies $o(\vec{t}) = o(\vec{u})$ if and only if it unifies $\vec{t}$ with $\vec{u}$. □

3.5 Case $M(\vec{x}) = o(\vec{u})$

We need to make the distinction whether it is a *simple* term pair or not, in order to avoid the algorithm to loop.

Definition 19. A term pair $t = u$ is called *simple* it is of the shape $M(\vec{x}) = o(N_1(\vec{y_1}), \ldots, N_n(\vec{y_n}))$, where

- $\vec{x}$, $\vec{y_1}$, $\ldots$, $\vec{y_n}$ are lists of variables and $\vec{x}$ are distinct;
- $\forall i, N_i \neq M$;
- o is a binding operation of the signature;
- the free variables in u are included in $\vec{x}$.

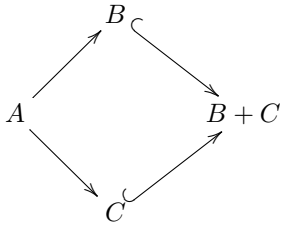
$$\frac{o \text{ has binding arity } (n_1, \ldots, n_p) \qquad M(\vec{x}) = o(\vec{N}) \text{ is a simple term pair}}{\Gamma \vdash M(\vec{x}) =_q o(\vec{N}) \Rightarrow M(1, \ldots, m) \mapsto o(\vec{N})[x_i \mapsto i] \dashv \Gamma \backslash \{M\}}$$

$$\frac{\begin{array}{c} o \text{ has binding arity } (n_1, \ldots, n_p) \qquad M(\vec{x}) = o(\vec{u}) \text{ is not a simple term pair} \qquad \forall i, u_i \neq M(\ldots) \\ \Gamma \backslash \{M\}, \vec{N} : m + \vec{n} \vdash N_1(0, \ldots, n_1 - 1, x_1 + n_1, \ldots, x_m + n_1) =_{q+n_1} u_1[M(1, \ldots m) \mapsto o(\vec{N})], \cdots \Rightarrow \sigma \dashv \Delta \end{array}}{\Gamma \vdash M(\vec{x}) =_q o(\vec{u}) \Rightarrow \sigma \circ M(1, \ldots m) \mapsto o(\vec{N}) \dashv \Delta}$$

Lemma 20. *The above rules are sound.*

Proof. If there is a unifier, then M can't appear in $\vec{u}$, so the condition $u_i \neq M(\ldots)$ is safe in both rules. Let us write Γ as $M : m, \Gamma'$, the coproduct of ym and Γ'. The term $M(\vec{x})$ corresponds to the composition of $yq \to T(ym)$ selecting $M(\vec{x})$ with the coproduct injection $T(ym) \hookrightarrow T(\Gamma)$.

In case of a simple term (first rule above), M does not appear in $o(\vec{u})$, the corresponding morphism $yq \to T(\Gamma)$ factors through the coproduct injection $T(\Gamma') \hookrightarrow T(\Gamma)$. In other words, we want to compute the coequaliser (in Kl_T) of



with $A = yq$, $B = ym$ and $C = \Gamma'$. This is equivalent to the pushout of $B \leftarrow A \to C$. Moreover, the free variable condition $fv(o(\vec{u})) \subset \vec{x}$ ensures that there exists $u' \in T(\Gamma')_m$ such that $u'[\vec{x}] = o(\vec{u})$, by choosing $u' = o(\vec{u})[x_i \mapsto i]$. Thus,

$o(\vec{u})$ can be decomposed as the composition $yq \xrightarrow{M(\vec{x})} T(ym) \xrightarrow{u'} T(\Gamma') \hookrightarrow T(\Gamma)$, and we want to compute the pushout

$$\begin{array}{ccc} yq & \xrightarrow{M(\vec{x})} & T(ym) \\ {\scriptstyle M(\vec{x})}\downarrow & & \\ T(ym) & & \\ {\scriptstyle u'}\downarrow & & \\ T(\Gamma') & & \end{array}$$

As we prove in the appendix (Lemma 28), since $\vec{x}$ is injective, the induced morphism $yq \xrightarrow{M(\vec{x})} T(ym)$ is epimorphic and thus we can eliminate it in the diagram, and the above pushout is simply given by $T(ym) \xrightarrow{u'} T(\Gamma') \xleftarrow{id} T(\Gamma')$, so that the coequaliser, as expected, substitute $M : m$ with u' and preserve the other metavariables.

Let us now tackle the other rule, by showing that the category of unifiers for the premise is equivalent to the category of unifiers for the conclusion. The morphism $yq \to T(\Gamma)$ corresponding to $o(\vec{u})$ factors through $T(\Gamma)^{(\vec{n})} \xrightarrow{o} T(\Gamma)$, where $T(\Gamma)_k^{(\vec{n})} = \prod_i T(\Gamma)_{k+n_i}$. A unifier $ym \to T(\Delta) \leftarrow \Gamma'$ must maps M to some $o(\dots)$, so that $ym \to T(\Delta)$ factors as $ym \to T(\Delta)^{(\vec{n})} \xrightarrow{o} T(\Delta)$. In other words, a unifier is a family Δ with morphisms $h : \Gamma' \to T(\Delta)$ and $g : ym \to T(\Delta)^{(\vec{n})}$ such that the following diagram commutes, where $f = [o \circ g, h] : \Gamma \to T(\Delta)$

$$\begin{array}{ccccc} yq & \xrightarrow{M(\vec{x})} & T(ym) & \xrightarrow{g^*} & T(\Delta)^{(\vec{n})} \\ {\scriptstyle u}\downarrow & & & & \\ T(\Gamma)^{(\vec{n})} & & & & \downarrow{\scriptstyle o} \\ {\scriptstyle o}\downarrow & & & & \\ T(\Gamma) & & \xrightarrow[f^*]{} & & T(\Delta) \end{array}$$

Now, since f^* is a T-algebra morphism, commutation of the previous diagram is equivalent to commutation of the following one:

$$\begin{array}{ccccc} yq & \xrightarrow{M(\vec{x})} & T(ym) & \xrightarrow{g^*} & T(\Delta)^{(\vec{n})} \\ {\scriptstyle u}\downarrow & & & & \\ T(\Gamma)^{(\vec{n})} & & & & \downarrow{\scriptstyle o} \\ {\scriptstyle f^{*(\vec{n})}}\downarrow & & & & \\ T(\Delta)^{(\vec{n})} & & \xrightarrow[o]{} & & T(\Delta) \end{array}$$

Since o is injective, this is equivalent to commutation of the following diagram.

$$\begin{array}{ccc} yq & \xrightarrow{M(\vec{x})} & T(ym) \\ {\scriptstyle u}\downarrow & & \downarrow{\scriptstyle g^*} \\ T(\Gamma)^{(\vec{n})} & \xrightarrow[f^{*(\vec{n})}]{} & T(\Delta)^{(\vec{n})} \end{array}$$

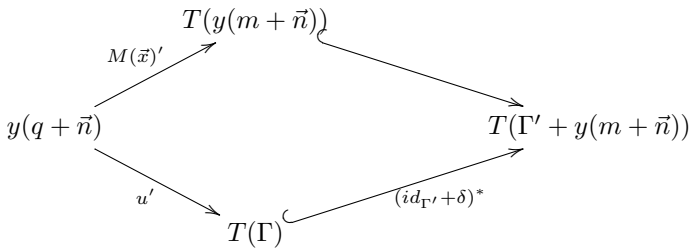
Now, a morphism $u : yk \to Y^{(\vec{n})}$ is equivalently given by a morphism $u' : y(k+\vec{n}) \to Y$, where $y(k+\vec{n})$ denotes the coproduct $\coprod_i y(k+n_i)$. Note that u can be retrieved from u' as the composition $yk \xrightarrow{d} y(k+\vec{n})^{(\vec{n})} \xrightarrow{u'^{(\vec{n})}} Y^{(\vec{n})}$. Thus, a unifier is given by a family Δ together with Kleisli morphisms $g' : y(m+\vec{n}) \to T(\Delta)$ and $h : \Gamma' \to T(\Delta)$ such that the following diagram commutes.

$$\begin{array}{ccc} y(q+\vec{n}) & \xrightarrow{M(\vec{x})'} & T(y(m+\vec{n})) \\ {\scriptstyle u'}\downarrow & & \downarrow{\scriptstyle g'^*} \\ T(\Gamma) & \xrightarrow[[o \circ g'^{(\vec{n})} \circ d, h]^*]{} & T(\Delta) \end{array}$$

Now, let us look at what a unifier is for the premise

$$\Gamma\backslash\{M\}, \vec{N} : m+\vec{n} \vdash N_1(0, \ldots, n_1-1, x_1+n_1, \ldots, x_m+n_1) =_{q+n_1} u_1[M(1, \ldots m) \mapsto o(\vec{N})], \cdots \Rightarrow \sigma \dashv \Delta$$

It is equivalently given as a cocone over the coequaliser diagram



where $\delta : ym \to T(y(m+\vec{n}))$ is given by $ym \xrightarrow{d} y(m+\vec{n})^{(\vec{n})} \to T(y(m+\vec{n})) \xrightarrow{o} T(y(m+\vec{n}))$. TODO: check that the category of unifiers are isomorphic. □

3.6 Case $M(\vec{x}) = x_i$

\vec{n}

$$\frac{}{\Gamma \vdash M(\vec{x}) =_q x_i : M(1, \ldots, m) \mapsto i \dashv \Gamma\backslash\{M\}}$$

A symmetric rule must be introduced as well.

Soundness is straightforward.

3.7 Termination

Proposition 21. *There is no infinite derivation tree of* $\Gamma \vdash \vec{t} =_{\vec{n}} \vec{u} \Rightarrow \sigma \dashv \Delta$.

Proof. TODO (Not sure how). □

References

[1] Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. Monads need not be endofunctors. *Logical Methods in Computer Science*, 11(1), 2015.

[2] Peio Borthelle, Tom Hirschowitz, and Ambroise Lafont. A cellular Howe theorem. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *Proc. 35th ACM/IEEE Symposium on Logic in Computer Science* ACM, 2020.

[3] Marcelo Fiore and Dmitrij Szamozvancev. Formal metatheory of second-order abstract syntax. *Proceedings of the ACM on Programming Languages*, 6(POPL), 2022.

[4] Saunders Mac Lane. *Categories for the Working Mathematician.* Number 5 in Graduate Texts in Mathematics. Springer, 2nd edition, 1998.

A Free T-algebras preserve pullbacks

In this section we show that renaming turn free T-algebras into functors $\mathbb{F} \to Set$ preserving pullbacks. In fact, we are going to show it for the initial T-algebra, since a free T-algebra on $X \in Set^{\mathbb{N}}$ is equivalently the initial Σ'-monoid, for Σ' the binding signature extending Σ with first order operations as specified by X.

Let us first explain where renaming comes from. Every binding signature Σ induces an endofunctor on $Set^{\mathbb{N}}$ that we still denote by Σ. For example, in the case of λ-calculus, $\Sigma(X)_n = X_n \times X_n + X_{n+1}$.

Lemma 22. *Denoting* Σ^* *the free monad* $\Sigma^* X = \mu Z.X + \Sigma(Z)$ *induced by a binding signature* Σ*, the initial* Σ*-monoid is* $\Sigma^* I$.

Proof. See [2, Theorem 2.15]. □

Lemma 23. *Given a binding signature* Σ*, the free monad* Σ^* *lifts to the category* $Set^{\mathbb{F}}$*. Denoting* $\underline{I}$ *the canonical embedding* $\mathbb{F} \to Set$*, renaming on* $\Sigma^* I$ *is given by the renaming structure on* $\Sigma^* \underline{I}$.

Proof. Renaming on $\Sigma^* I$ is induced by substitution. Substitution on $\Sigma^* I$ induced by a monoid structure on them (for a monoidal structure on $Set^{\mathbb{F}}$ or $Set^{\mathbb{N}}$, as in [1]) induced by a strength $\Sigma(A) \otimes B \to \Sigma(A \otimes B)$.

TODO □

What remains to show is that this renaming preserves pullbacks. Let us reason on the category $Set^{\mathbb{F}}$ on which Σ^* lifts, as we just argued. Now, $\Sigma^* I$ is computed as the initial algebra of $H : Set^{\mathbb{F}} \to Set^{\mathbb{F}}$ mapping X to $I + \Sigma(X)$. This means that $\Sigma^* I$ is the colimit over the initial ω-chain

$$\emptyset \to H(\emptyset) \to \cdots \to H..H(..(\emptyset..)) \to \ldots$$

Since pullback preserving functors are preserved by filtered colimits (as filtered colimits commute with finite limits), it is enough to show that each functor $H^{\circ n}(\emptyset)$ preseves pullbacks, which again amounts to showing that H preserves functors preservings pullbacks.

Lemma 24. *The endofunctor $H : Set^{\mathbb{F}} \to Set^{\mathbb{F}}$ maps functors preserving pullbacks to functors preserving pullbacks.*

Proof. Let F be a functor $\mathbb{F} \to Set$ preserving pullbacks. Let

$$\begin{array}{ccc} A & \longrightarrow & B \\ \downarrow & & \downarrow \\ C & \longrightarrow & D \end{array}$$

be a pullback in $\mathbb{F}$. Let us show that the following square is a pullback.

$$\begin{array}{ccc} H(F)_A & \longrightarrow & H(F)_B \\ \downarrow & & \downarrow \\ H(F)_C & \longrightarrow & H(F)_D \end{array}$$

Since pullbacks commute with coproducts in Set, this is enough to show that the following square is a pullback

$$\begin{array}{ccc} \Sigma(F)_A & \longrightarrow & \Sigma(F)_B \\ \downarrow & & \downarrow \\ \Sigma(F)_C & \longrightarrow & \Sigma(F)_D \end{array}$$

This follows from the fact that $\Sigma(F)$ is isomorphic to some $\coprod_i F^{(\vec{n_i})}$, and coproducts commute with connected limits. □

Notation 25. Let $Set_p^{\mathbb{F}}$ be the full subcategory of functors preserving pullbacks.

Lemma 26. *The functor $\mathbb{N} \xrightarrow{y} Set^{\mathbb{N}} \to Kl_T$ extends to a functor $\mathbf{y} : \mathbb{F}^o \to Kl_T$ mapping $f : n \to m$ to*

$$ym \to (Tyn)_n \cdot ym \xrightarrow{(Tyn)_f} (Tyn)_m \cdot ym \to Tyn$$

Proof. There is a family of adjunction $Set(X, R_m Y) \simeq Kl_T(L_m X, Y)$ parameterised by $m \in \mathbb{N}$, where $L_m X = X \cdot ym$ and $R_m Y = TY_m$. By , L_m Since R_m is functorial in $m \in \mathbb{F}$, by [4, Theorem IV.7.3] L_m is functorial in $\mathbb{F}^o$ with action on $f : n \to m$ given by

$$L_m \to L_m R_n L_n \xrightarrow{L_m R_f L_n} L_m R_m L_n \to L_n$$

In particular, $L_m 1 \simeq ym$ is contravariantly functorial in $m \in \mathbb{F}$. □

The rest of this section is devoted to the proof of the following statement.

Lemma 27. $\mathbf{y}^o : \mathbb{F} \to Kl_T^o$ *preserves pullbacks, hence finite connected limits.*

Proof. We have a natural isomorphism $Kl_T(\mathbf{y}m, X) \simeq T(X)_m \simeq Set_p^{\mathbb{F}}(Jm, TX)$, where J is the yoneda embedding in $\mathbb{F}^o \to Set^{\mathbb{F}}$: $Jn_m = m^n$. It follows that $\mathbf{y}$ preserves any colimit that J preserves. Now, since J is the yoneda embedding, J preserves pullbacks, as we restrict to the category of presheaves preserving pullbacks. □

Corollary 28. *Let $f : p \to q$ be an injective map. Then, the induced map $\mathbf{y}f : yq \to T(yp)$ is epimorphic.*

Proof. The proof follows from Lemma 27 by noting that $f : A \to B$ is monomorphic if and only if the following square is a pullback.

$$\begin{array}{ccc} A & \xrightarrow{id} & A \\ {\scriptstyle id}\downarrow & & \downarrow{\scriptstyle f} \\ A & \xrightarrow[f]{} & B \end{array}$$

□