Pattern unification in second-order abstract syntax

June 8, 2022

We give a correctness proof for pattern unification for second-order abstract languages. Other envisionned applications of our abstract categorical proof are unification for linear or polymorphic syntax (Section 8). In Section 1, we propose a general categorical setting in which pattern unification applies, with the nominal sets in mind. In this case, a second-order syntax with metavariables applied to distinct variables (according to the pattern restriction) corresponds to a free monad applied a coproduct of representable presheaves.

In Section 2, we state our main result that justifies pattern unification algorithms. Then we tackle the proof algorithm, starting with the pruning phase (Section 4), the coequalising phase (Section 5), the occur-check phase (Section 6), and finally we justify termination (Section 7).

Notation 1. Given a natural number n we denote by $\underline{n}$ the set $\{0, \ldots, n-1\}$.

1 Categorical setting

Here we describe our setting, based on the example nominal sets. We explicitly refer to these assumptions in the document when they are used.

Assumption 2. *We assume given a locally presentable category $\mathcal{C}$.*

Presentability ensures (in particular) that $\mathcal{C}$ is bicomplete.

Remark 3. We need cocompleteness so that we can compute free monads of a finitary endofunctor. Completeness ensures that this free monad is algebraically free (TODO check that this is really needed).

Question 4. *is bicompleteness enough?*

Example 5. Consider Nom the category of nominal sets, as functors $\mathbb{F}_m \to Set$ preserving pullbacks, where $\mathbb{F}_m$ is the category of finite cardinals and injections between them.

Assumption 6. *$\mathcal{C}$ is extensive.*

This is useful for coproduct properties (e.g., coproduct injections $A \hookrightarrow A + B \hookleftarrow B$ are monomorphic).

Example 7. As a topos, Nom is extensive.

Assumption 8. *We assume given a full subcategory embedding $D \xrightarrow{K} \mathcal{C}$.*

Example 9. Representable presheaves $\hom_{\mathbb{F}_m}(n, -) : \mathbb{F}_m \to Set$ preserve limits and thus induce nominal sets. Thus, the yoneda embedding $\mathbb{F}_m^{op} \to [\mathbb{F}_m, Set]$ factors as $\mathbb{F}_m^{op} \xrightarrow{K} \mathrm{Nom} \hookrightarrow [\mathbb{F}_m, Set]$

Definition 10. We denote by $D^+ \xrightarrow{K^+} \mathcal{C}$ the full subcategory of $\mathcal{C}$ consisting of finite coproducts of objects of D.

We will be interested in coequalisers in D^+.

Assumption 11. *D has finite connected colimits and K preserve them.*

This is to deal with the case $M(\vec{x}) = N(\vec{y})$. Note that this statement is equivalent by replacing finite connected colimits with coequalisers and pushouts.

Example 12. In the case of nominal sets, the yoneda embedding $K : \mathbb{F}_m^{op} \to Nom$ preserves them because nominal sets are functors $\mathbb{F}_m \to Set$ preserving finite connected limits.

Remark 13. I think that the algorithm still works without this assumption: it is just that at some point we need to compute a colimit of elements of D, in particular in Section 5.2. We already know that such a colimit exists in $\mathcal{C}$, but in concrete examples, it is better to know that we can compute them in D because this category is simpler (TODO: think more about it).

Assumption 14. *Morphisms in D are epimorphisms.*

In the case of nominal sets, this comes from the fact that we restrict to monomorphisms. Note that the image by K of an epimorphism is epimorphic by 11.

Assumption 15. *For each object $d \in D$, $\hom_{\mathcal{C}}(Kd, -)$ preserves coproducts (i.e., d is connected) and filtered colimits (TODO: do we really need the latter?).*

Again, this is useful to know to factor $Kd \to A + B$ as $Kd \to A$ or $Kd \to B$.

Example 16. Because pullbacks commute with coproducts and filtered colimits in sets, such colimits in Nom are computed pointwise and thus representable presheaves have the required property.

Assumption 17. *We assume given a finitary endofunctor F on $\mathcal{C}$. We denote by T the generated free monad F^*.*

Example 18. Consider the endofunctor on Nom corresponding to λ-calculus: $F(X) = I + X \times X + X^I$, where I is the representable presheaf $y1$. Note that $X_n^I = X_{n+1}$.

Assumption 19. *$F(X)$ is of the shape $I + \coprod_i R_i(X)$, where*

- *I is an object of $\mathcal{C}$;*
- *R_i is right adjoint to some L_i;*
- *$L_i K = \coprod_{j \in J} K L'_{i,j}$ for some finite family of endofunctors $(L'_{i,j})_j$ on D.*

Example 20. Continuing Example 18, $X \mapsto X \times X$ is right adjoint to $X \mapsto X + X$. Moreover, $X \mapsto X^I$ is right adjoint to $X \mapsto X \times I$. If X is a representable presheaf yn, then $yn \times I = yn \times y1 \simeq y(n+1)$. Therefore, $L'n = n+1$.

2 Main result

Our main result is that a coequaliser diagram in Kl_T either has no unifier, either can be decomposed further. An additionnal argument is necessary to ensure termination of the decomposition, for coequaliser diagrams selecting objects in D^+.

But first let us rephrase the statement.

Definition 21. Given a category E, let E^* be E extended freely with a terminal object.

Remark 22. Adding a terminal object results in adding a terminal cocone to all diagrams.

As a consequence, we have a following lemma.

Lemma 23. *Let J be a diagram in a category E. The following are equivalent:*

1. *J has a colimit has long as its category of cocones is not empty.*
2. *J has a colimit in E^*.*

Thus, we are going to work in Kl_T^* rather than Kl_T.
The following result is also useful.

Lemma 24. *Given a category E, the canonical functor $E \to E^*$ creates colimits.*

This has some consequences:

1. whenever the colimit in Kl_T^* is not the terminal object, it is also a colimit in Kl_T;
2. existing colimits in Kl_T are also colimits in Kl_T^*;
3. in particular, coproducts in Kl_T (which are computed in $\mathcal{C}$) are also coproducts in Kl_T^*.

Notation 25. We denote by $\top$ the terminal object and by ! any terminal morphism.

3 Notations

We denote the identity morphism at an object x by 1_x.

If $(g_i : A_i \to B)_{i\in I}$ is a family of arrows, we denote by $[g_i] : \coprod_{i\in I} A_i \to B$ the induced coproduct pairing.

Coproduct injections $A_i \to \coprod_{i\in I} A_i$ are typically denoted by in_i.

Given an adjunction $L \dashv R$ and a morphism $f : A \to RB$, we denote by $f^* : LA \to B$ its transpose, and similarly, if $g : LA \to B$, then $g^* : A \to RB$. In particular, a Kleisli morphism $f : A \to TB$ induces a morphism $f^* : TA \to TB$ through the adjunction between Kl_T and $\mathcal{C}$.

We denote the Kleisli composition of $f : A \to TB$ and $g : TB \to TC$ by $f[g] = g^* \circ f$.

4 Pruning phase

Here we want to compute a pushout diagram in Kl^*_T, where one branch is a coproduct of free morphisms.

$$\begin{tikzcd}
\coprod_i KA_i \arrow[r, "\coprod_i Kf_i"] \arrow[d, "{[g_i]}"'] & \coprod_i KB_i \arrow[r, "\eta"] & T\coprod_i KB_i \arrow[d, "{[u_i]}"] \\
X \arrow[rr, "\sigma"'] & & Z
\end{tikzcd}$$

where $g_i : KA_i \to X$ and $u_i \in \hom_{Kl^*_T}(KB_i, Z)$. We denote such a situation by

$$X \vdash f_1 := g_1, f_2 := g_2, \dots \Rightarrow u_1, u_2, \dots; \sigma \dashv Z$$

abbreviated as

$$X \vdash \vec{f} := \vec{g} \Rightarrow \vec{u}; \sigma \dashv Z$$

or even

$$X \vdash (f_i)_i := g \Rightarrow u; \sigma \dashv Z$$

with $g = [g_i]$ and $u = [u_i]$.

The simplest case is when the coproduct is empty: then, the pushout is X.

$$\frac{}{X \vdash \vec{()} := \vec{()} \Rightarrow \vec{()}; 1_X \dashv X}$$

Another simple case is when $X = \top$. Then, the pushout is the terminal cocone. Thus we have the rule

$$\frac{}{\top \vdash \vec{f} := \vec{g} \Rightarrow \vec{!}; ! \dashv \top}$$

The pushout can be decomposed into smaller components.

$$\frac{X \vdash \vec{f} := \vec{g} \Rightarrow \vec{u}; \sigma \dashv Z \qquad Z \vdash \vec{f'} := \vec{g'}[\sigma] \Rightarrow \vec{u'}; \sigma' \dashv Z'}{X \vdash \vec{f}, \vec{f'} := \vec{g}, \vec{g'} \Rightarrow \vec{u}[\sigma], \vec{u'}; \sigma \dashv Z}$$

This follows from the following general lemma.

Lemma 26. *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 $A_2 \rightrightarrows B \to C$.*

Thanks to the previous rule, we can focus on the case where the coproduct is the singleton (since we focus on finite coproducts of elements of D). Thus, we want to compute the pushout

$$\begin{tikzcd} KA \arrow[r, "Kf"] \arrow[d, "g"'] & KB \arrow[r, "\eta"] & TKB \\ TC & & \end{tikzcd}$$

Since $TC \simeq I + C + \coprod_i R_i TC$ (Assumption 19) and KA is connected (Assumption 15), $KA \to TC$ factors through one of the following coproduct injections:

- $in_I : I \hookrightarrow TC$ (variable case)
- $\eta : C \hookrightarrow TC$ (metavariable case)
- $in_i : R_i TC \hookrightarrow TC$ (operation case)

In the next subsections, we discuss the different cases.

4.1 Case $KA \hookrightarrow I$

A cocone in Kl_T is given by an object Y with morphisms $KB \to TY \leftarrow C$ such that the following diagram commutes.

$$\begin{tikzcd} KA \arrow[r, "Kf"] \arrow[d, "g"'] & KB \arrow[d] \\ I \arrow[r] & TY \end{tikzcd}$$

Since KB is connected (Assumption 15), by extensivity (Assumption 6) and definition of T (Assumption 19), $KB \to TY$ factors through $I \hookrightarrow TY$. Therefore, a unifier is equivalently an object $Y \in D^+$ with a morphism $KC \to TY$ and a morphism $u : KB \to I$ that makes the following diagram commute.

$$\begin{tikzcd} KA \arrow[rr, "Kf"] \arrow[dr, "g"'] & & KB \arrow[dl, "u"] \\ & I & \end{tikzcd}$$

We already know that such a morphism $KB \to I$ is unique, as the top morphism is epimorphic by Assumptions 11 and 14. Therefore, we have the following rule in case there exists such a u.

$$\frac{g = u \circ Kf}{C \vdash f := in_I \circ g \Rightarrow in_I \circ u; 1_C \dashv C}$$

and the following rule in case no such morphism exists (this happens concretely for $M(x) := y$ when $y \neq x$).

$$\frac{\forall u : KB \to I, g \neq u \circ Kf}{C \vdash f := in_I \circ g \Rightarrow !; ! \dashv \top}$$

Remark 27. If Kf is effective (e.g., if $\mathcal{C}$ is a topos, as is the case of nominal sets), then the existence of such a u is equivalent to g coequalising the kernel of Kf.

4.2 Case $KA \hookrightarrow C$

We want to compute the pushout of free morphisms

$$\begin{tikzcd} KA \arrow[r, "Kf"] \arrow[d, "g"'] & KB \arrow[r, "\eta"] & TKB \\ C \arrow[d, "\eta"'] & & \\ TC & & \end{tikzcd}$$

Since the functor $\mathcal{C} \to Kl_T$ is left adjoint, the pushout can be computed in $\mathcal{C}$, and it exists by Assumption 2. Therefore, we have the rule

$$\frac{\begin{tikzcd} KA \arrow[r, "Kf"] \arrow[d, "g"'] & KB \arrow[d, "u"] \\ C \arrow[r, "\sigma"'] & D \end{tikzcd} \text{ is a pushout}}{C \vdash f := \eta \circ g \Rightarrow \eta \circ u; T\sigma \dashv D}$$

4.3 Case $KA \hookrightarrow R_i TC$

We want to compute the pushout

$$\begin{tikzcd} KA \arrow[r, "Kf"] \arrow[d, "g"'] & KB \arrow[r, "\eta"] & TKB \\ R_i TC \arrow[d, "in_i"'] & & \\ TC & & \end{tikzcd}$$

A cocone in Kl_T is given by an object Y with morphisms $KB \to TY \leftarrow C$ such that the following diagram commutes.

$$\begin{tikzcd} KA \arrow[rr, "Kf"] \arrow[d, "g"'] & & KB \arrow[d] \\ R_iTC \arrow[r] & R_iTY \arrow[r] & TY \end{tikzcd}$$

Since KB is connected (Assumption 15), by extensivity (Assumption 6) and definition of T (Assumption 19), $KB \to TY$ factors through $R_iTY \hookrightarrow TY$. Since $R_iTY \to TY$ is monomorphic (as a coproduct injection, again by Assumption 6), a cocone in Kl_T is given by an object Y with morphisms $C \to TY$ and $KB \to R_iTY$ such that the following diagram commutes.

$$\begin{tikzcd} KA \arrow[r, "Kf"] \arrow[d, "g"'] & KB \arrow[d] \\ R_iTC \arrow[r] & R_iTY \end{tikzcd}$$

Since R_i has a left adjoint L_i such that $L_iK = \int^j KL'_{i,j}$ (Assumption 19), this is equivalent to making the following diagram commute.

$$\begin{tikzcd} \coprod_j KL'_{i,j}A \arrow[r, "\coprod_j KL'_{i,j}f"] \arrow[d, "g^*"'] & \coprod_j KL'_{i,j}B \arrow[d, "u"] \\ TC \arrow[r, "\sigma"'] & TY \end{tikzcd}$$

Thus, this justifies the following rule

$$\frac{C \vdash (L'_{i,j}f)_j := g^* \Rightarrow u; \sigma \dashv Y}{C \vdash f := in_i \circ g \Rightarrow in_i \circ u^*; \sigma \dashv Y}$$

5 Coequalising phase

Here we want to compute a coequalising diagram in Kl_T^*, where the domain is in D^+.

$$\begin{tikzcd} \coprod_i KA_i \arrow[r, shift left, "{[t_i]}"] \arrow[r, shift right, "{[u_i]}"'] & \Gamma \arrow[r, "\sigma"] & \Delta \end{tikzcd}$$

where $g_i : KA_i \to X$ and $u_i \in \hom_{Kl_T^*}(KB_i, Z)$. We denote such a situation by

$$\Gamma \vdash t_1 =_{A_1} u_1, \ldots t_p =_{A_p} u_p \Rightarrow \sigma \dashv \Delta$$

that we sometimes abbreviate as

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

The simplest case is when the coproduct is empty: then, the coequaliser is Γ

$$\frac{}{\Gamma \vdash () \Rightarrow 1_\Gamma \dashv \Gamma}$$

Another simple case is when $\Gamma = \top$. Then, the pushout is the terminal cocone. Thus we have the rule

$$\frac{}{\top \vdash \vec{t} =_{\vec{A}} \vec{u} \Rightarrow ! \dashv \top}$$

Again, thanks to Lemma 26, such a coequaliser can be decomposed into smaller components.

$$\frac{\Gamma \vdash t_0 =_{n_0} u_0 \Rightarrow \sigma_0 \dashv \Delta_1 \qquad \Delta_1 \vdash \vec{t}[\sigma_0] =_{\vec{n}} \vec{u}[\sigma_0] \Rightarrow \sigma \dashv \Delta_2}{\Gamma \vdash t_0 =_{n_0} u_0, \vec{t} =_{\vec{n}} \vec{u} \Rightarrow \sigma_0[\sigma] \dashv \Delta_2}$$

Thanks to the previous rules, we can focus on the case where the coproduct is a singleton (since we focus on finite coproducts of elements of D), and $\Gamma = TC$. Thus, we want to compute the coequaliser

$$KA \overset{t}{\underset{u}{\rightrightarrows}} TC$$

Since $TC \simeq I + C + \coprod_i R_i TC$ (Assumption 19) and KA is connected (Assumption 15), $t, u : KA \to TC$ factor through one of the following coproduct injections:

- $in_I : I \hookrightarrow TC$ (variables)
- $\eta : C \hookrightarrow TC$ (metavariables)
- $in_i : R_i TC \hookrightarrow TC$ (operations)

In the next subsections, we discuss the different cases.

- $\eta = \ldots$, or $M(\vec{x}) = ...$, in case of a successful occur-check, in Section 5.1;
- $\eta = \eta$, or $M(\vec{x}) = N(\vec{y})$, in Section 5.2, which is partly redundant with the previous case
- $in_i = in_i$, or $o(\vec{t}) = o(\vec{u})$, in Section 5.3;
- $in_I = in_I$, or $x = y$, in Section 5.4.
-

Let us mention schematically some other cases, which can never be unified in Kl_T and thus are solved using $\top$.

- $in_i = in_{i'}$ with $i \neq i'$ (in the examples, $o(\vec{t}) = o'(\vec{u})$ when $o \neq o'$)
- $in_i = in_I$ (in the examples, $o(\vec{t}) = x$)
- $\eta = \dots$, or $M(\vec{x}) = u$, with failing occur-check (Section 6), i.e., when M appears deep in u.

5.1 Successful occur-check

WIP

The occur-check phase is described in more details in Section 6. Here, we assume that

1. $C \simeq KB + C'$ for some B, C';
2. $t : KA \to TC$ factors as $KA \xrightarrow{f} KB \xhookrightarrow{in_B} TC \simeq KB + \dots$;
3. $u : KA \to TC$ factors as $KA \xrightarrow{g} TC' \xhookrightarrow{in_{TC'}} TC$.

In the examples, the second condition means that t is a metavariable, and the last condition means that this metavariable does not occur in u, i.e., the occur-check is successful.

Thus, the coequaliser

$$\begin{tikzcd} KA \arrow[r, shift left, "t"] \arrow[r, shift right, "u"'] & TC \end{tikzcd}$$

is a coequaliser (in Kl_T^*) of the shape

$$\begin{tikzcd} & Y \arrow[dr] & \\ X \arrow[ur] \arrow[dr] & & Y+Z \\ & Z \arrow[ur] & \end{tikzcd},$$

with $X = KA$, $Y = TKB$ and $Z = TC'$

Remark 28. There is a canonical isomorphism between the category of cocones over such a coequaliser diagram and the category of cocones over the pushout diagram $Y \leftarrow X \to Z$ exists.

Therefore, computing this coequaliser amounts to computing the pushout. We are thus in the situation of the pruning phase, and we can justify the rule

$$\frac{C' \vdash f := g \Rightarrow v; \sigma \dashv Z}{KB + C' \vdash in_B \circ f = in_{TC'} \circ g \Rightarrow [v, \sigma] \dashv Z}$$

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

Here we are in the situation where $t, u : KA \to TC$ factor as $t', u' : KA \to C$ through $\eta : C \to TC$. Note that since postcomposition with η is precisely the left adjoint (and thus cocontinuous) functor from $\mathcal{C}$ to Kl_T, it is enough to compute the coequaliser in $\mathcal{C}$ and then precompose it with η.

We assume the following

- $C \simeq \coprod_{i \in \{a,b\}} KB_i + C'$, where a, b may not be distinct (in practice C is in D^+),
- $t', u' : KA \to C$ factor respectively as $KB_a \to C$ and $KB_b \to C$.

Note that if a and b are distinct, then the case we describe is redundant with some specific case of the pruning phase (Section 4.2).

We want to compute the coequaliser of the free algebra morphisms.

$$\begin{tikzcd}
 & KB_a \arrow[dr] & \\
KA \arrow[ur, "Kf"] \arrow[dr, "Kg"'] & & \coprod_{i \in \{a,b\}} KB_i + C' \\
 & KB_b \arrow[ur] &
\end{tikzcd}$$

Lemma 29. *The coequaliser of a diagram of this shape, in any category which has the involved colimits, is $X + C'$, where X is the coequaliser or Kf and Kg if $a = b$, or the pushout of Kf and Kg, otherwise.*

Thanks to Assumption 11, we know that such a colimit exists.

All the arguments in this section justify the following rule, whose premise can always be satisfied.

$$\frac{\begin{tikzcd}
 & KB_a \arrow[dr] & & \\
KA \arrow[ur, "Kf"] \arrow[dr, "Kg"'] & & \coprod_{i \in \{a,b\}} KB_i + C' \arrow[r, "\sigma"] & X \text{ is a pushout} \\
 & KB_b \arrow[ur] & &
\end{tikzcd}}{\coprod_{i \in \{a,b\}} KB_i + C' \vdash \eta \circ in_a \circ Kf = \eta \circ in_b \circ Kf \Rightarrow \eta \circ \sigma \dashv X}$$

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

Here we assume that $t, u : KA \to TC$ factors as $t', u' : KA \to R_i TC$ through $R_i TC \hookrightarrow TC$. A cocone in Kl_T is given by an object Y with a morphism

$C \xrightarrow{\sigma} TY$ such that the following diagram commutes.

$$\begin{tikzcd} KA \arrow[r, "t'"] \arrow[d, "u'"'] & R_iTC \arrow[r, "R_i\sigma^*"] & R_iTY \arrow[d, "in_i"] \\ R_iTC \arrow[r, "R_i\sigma^*"'] & R_iTY \arrow[r, "in_i"'] & TY \end{tikzcd}$$

Since $R_iTY \to TY$ is monomorphic (as a coproduct injection, by extensivity, Assumption 6), the above commutation is equivalent to commutation of the following diagram.

$$\begin{tikzcd} KA \arrow[r, "t'"] \arrow[d, "u'"'] & R_iTC \arrow[d, "R_i\sigma^*"] \\ R_iTC \arrow[r, "R_i\sigma^*"'] & R_iTY \end{tikzcd}$$

Since R_i has a left adjoint L_i such that $L_iK = \int^j KL'_{i,j}$ (Assumption 19), this is equivalent to making the following diagram commute.

$$\begin{tikzcd} \coprod_j KL'_{i,j}A \arrow[r, "t'^*"] \arrow[d, "u'^*"'] & TC \arrow[d, "\sigma^*"] \\ TC \arrow[r, "\sigma^*"'] & TY \end{tikzcd}$$

Thus, this justifies the following rule

$$\frac{C \vdash t'^* = u'^* \Rightarrow \sigma \dashv Y}{C \vdash in_i \circ t' = in_i \circ u' \Rightarrow \sigma \dashv Y}$$

5.4 Case $x = y$

Here we assume that $t, u : KA \to TC$ factors as $t', u' : KA \to I$ through $I \hookrightarrow TC$. A coequalising cocone in Kl_T is an object D such that $I \hookrightarrow TD$ coequalises t' and u'. Since $I \hookrightarrow TD$ is monomorphic (by extensivity, Assumption 6), this implies that $t' = u'$. In other words, either $t' = u'$ and in this case the coequaliser is just TC, either $t' \neq u'$ and in this case there is no unifier in Kl_T. The first case is a particular instance of the following rule

$$\frac{}{C \vdash f = f \Rightarrow 1_C \dashv C}$$

The second case can be expressed with the following rule.

$$\frac{t' \neq u'}{C \vdash in_I \circ t' = in_I \circ u' \Rightarrow ! \dashv \top}$$

6 Occur-check

WIP

The occur-check allows to jump from the coequalising phase (Section 5) to the pruning phase (Section 4), whenever the metavariable appearing at the toplevel of the l.h.s does not appear in the r.h.s. On the other hand, if it appears on the r.h.s and is not top-level, then there is no unifier.

The challenge here is to define the algorithm recursively and prove it correct.

Let $J : \mathcal{C} \to \hat{D}$ mapping c to $\hom_{\mathcal{C}}(K-, c)$. Moreover, let $G : \hat{D} \to \hat{D}$ mapping P to $\coprod_i \prod_j P_{L_{i,j}-}$. Then, GJ is isomorphic to JF, i.e., the following square commutes up to isomorphism.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "F"] \arrow[d, "J"'] & \mathcal{C} \arrow[d, "J"] \\
\hat{D} \arrow[r, "G"'] & \hat{D}
\end{tikzcd}$$

Note that J has a left adjoint (since it is continuous), but more importantly, it preserves coproducts and filtered colimits colimits by Assumption 15. As a consequence, we the following square commutes up to isomorphism.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "T"] \arrow[d, "J"'] & \mathcal{C} \arrow[d, "J"] \\
\hat{D} \arrow[r, "G^*"'] & \hat{D}
\end{tikzcd}$$

Then, we can define the size of a morphism as a universal G-algebra morphism to the constant presheaf $\mathbb{N}$, or define the occur-check by induction as a G-algebra morphism from $\hom_{\mathcal{C}}(K-, T(B + C))$ to $\hom_{\mathcal{C}}(K-, TC) + 1$. .

7 Termination

TODO The usual termination argument should apply (but we need first to disambiguate the rules above). Indeed, because objects of D are connected (Assumption 15) in an extensive category (Assumption 6), we can define without ambiguity the *size* of a *context* (i.e., an element of D^+) as the length of the coproduct.

8 Applications

8.1 Nominal sets (untyped)

8.2 Nominal sets, simply-typed

8.3 Linear syntax

Take $\mathcal{C} = Set^{\mathbb{N}}$ and D the full subcategory of representable presheaves. Intuitively, given $X \in Set^{\mathbb{N}}$, the set X_n is the set of expressions with exactly n (distinct) variables. Then, we can consider the linear lambda-calculus, as an endofunctor on $Set^{\mathbb{N}}$ mapping X to $F(X)$ where $F(X)_n = y1 + \coprod_{p+q=n} X_p \times X_q + (n+1) \times X_{n+1}$. Note that we could also specify a non-linear binder by replacing $(n+1) \times X_{n+1}$ with $\coprod_{p>n} \begin{pmatrix} p \\ n \end{pmatrix} X_p$. We could also have a non linear application by replacing $\coprod_{p+q=n} X_p \times X_q$ with $X_n \times X_n$.

Then, $F^*(0)$ is the linear lambda-calculus. $F^*(yn)$ is the syntax of linear λ-calculus extended with one n-ary metavariable applied to n (distinct) variables.

8.4 Polymorphic syntax

WIP. Let $J^+ : \mathbb{F} \to Set$ denotes the functor mapping n to the $n + 1^{th}$ cardinal $\{0, \ldots, n\}$.

Let S be a monad on Set such that SX is the set of types taking free type variables in X.

We need to consider something like $Set^{J^+/SJ}$, but restricted to a full subcategory embedding so that the yoneda embedding preserve finite connected limits.