Pattern unification in second-order abstract syntax

July 25, 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 9). 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 3, we state our main result that justifies pattern unification algorithms. Then we tackle the proof algorithm, starting with the pruning phase (Section 5), the coequalising phase (Section 6), the occur-check phase (Section 7), and finally we justify termination (Section 8).

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

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 6.2. We already know that such a colimit exists in $\mathcal{C}$, but in concrete examples, but we need to compute them in D so that we can chain the coequalisers when unifying multiple terms.

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 $I + \coprod_i R_i(X) \times S_i$, where*

- *I is an object of $\mathcal{C}$;*
- *R_i is right adjoint to some L_i;*
- *S_i is a orthogonal to the class of all morphisms (i.e., given any span $S_i \leftarrow A \rightarrow B$, there exists a unique $B \rightarrow S_i$ completing the triangle) . Intuitively, S_i is the output type "dirac" (not used for untyped)*
-
- *$L_i K = \coprod_{j \in J} K L'_{i,j}$ 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, $L'n = n+1$.

This condition is too strong: lambda claculs extended with a constant does not satisfy this constraint (really? Isn't the terminal functor right adjoint to the initial functor? Otherwise, can't we put the constants in I?).

2 Examples of relevant categorical contexts

Typically, we work in a full subcategory of functors $C \rightarrow Set$ preserving pullbacks, such that morphisms in C are all monomorphisms. With some conditions, this is a topos and hence has a number of good properties. Then, K is the coyoneda embedding.

Let $(C_i)_i$ be a family of categories such that each C_i (except the last one, these properties are taken from the paragraph before conclusion in [3], the author says he will present more explictly his results in a later paper, but I couldn't find a reference):

- is boolean (meaning any subobject has a complement),
- has effective unions (not sure what it means)
- has finite limits
- has finite coproducts
- there is no infinite chain of proper subobjects
- all morphisms are monomorphisms

Denoting B_i the core of C_i, the left Kan extension along $B_i \rightarrow C_i$ induces a monad. According to Menni, one can prove that its Kleisli category is a Grothendieck topos. Moreover, by [1, Theorem 1.7, Lemma 1.8], this category is equivalent the full subcategory $\mathcal{P}_i$ of functors $C_i \rightarrow Set$ preserving pullbacks.

Thus, $\prod \mathcal{P}_i$ is also a Grothendieck topos (by Lemma 42) and is equivalent to the full subcategory of functors $\coprod_i C_i \rightarrow Set$ preserving pullbacks. Thus, it contains the representables presheaves and we consider for K the yoneda embedding.

3 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.

Here is our main result.

Theorem 26. *Let $Kl_{D^+}^*$ be the full subcategory of Kl_T^* consisting of objects of $D^+ \cup \{\top\}$. Then, $Kl_{D^+}^*$ has coequalisers and the inclusion $Kl_{D^+}^* \to Kl_T^*$ preserves them.*

In other words, any coequaliser diagram $A \rightrightarrows TB$ in Kl_T^* where A and B are in D^+ has a colimit $B \to TC$ where $C \in D^+ \cup \{\top\}$. Rephrasing it without the adjoined terminal object, this means that either such a coequaliser has a colimit which can be computed in $Kl_{D^+}^*$, either there is no cocone at all.

4 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 $f[g] = g^* \circ f$.

5 Pruning phase

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

$$\begin{array}{ccccc} \coprod_i KA_i & \xrightarrow{\coprod_i Kf_i} & \coprod_i KB_i & \xrightarrow{\eta} & T\coprod_i KB_i \\ {\scriptstyle [g_i]}\downarrow & & & & \downarrow{\scriptstyle [u_i]} \\ X & & \xrightarrow[\sigma]{} & & Z \end{array}$$

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

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

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 $u = [u_i]$. Could we use Reddy's syntax, to differentiate the input/output? We need to know what his syntax is the internal language of.

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 stepwise construction of coequalisers (Lemma 39).

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

$$\begin{array}{ccccc} KA & \xrightarrow{Kf} & KB & \xrightarrow{\eta} & TKB \\ {\scriptstyle g}\downarrow & & & & \\ TC & & & & \end{array}$$

Since $TC \simeq I + C + \coprod_i R_i TC \times S_i$ (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)
- $in_i : R_i TC \times S_i \hookrightarrow TC$ (operation case)

In the next subsections, we discuss the different cases.

5.1 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.

$$\begin{array}{ccc} KA & \xrightarrow{Kf} & KB \\ {\scriptstyle g}\downarrow & & \downarrow \\ I & \longrightarrow & TY \end{array}$$

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.

$$\begin{array}{ccccc} KA & & \xrightarrow{Kf} & & KB \\ & {\scriptstyle g}\searrow & & \swarrow{\scriptstyle u} & \\ & & I & & \end{array}$$

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.

5.2 Case $KA \hookrightarrow C$

We want to compute the pushout of free morphisms

$$\begin{array}{ccccc} KA & \xrightarrow{Kf} & KB & \xrightarrow{\eta} & TKB \\ {\scriptstyle g}\downarrow & & & & \\ C & & & & \\ {\scriptstyle \eta}\downarrow & & & & \\ TC & & & & \end{array}$$

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

$$\frac{\begin{array}{ccc} KA & \xrightarrow{Kf} & KB \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle u} \\ C & \xrightarrow[\sigma]{} & D \end{array} \text{ is a pushout}}{C \vdash f := \eta \circ g \Rightarrow \eta \circ u; T\sigma \dashv D}$$

5.3 Case $KA \hookrightarrow R_i TC \times S_i$

We want to compute the pushout

$$\begin{array}{ccccc} KA & \xrightarrow{Kf} & KB & \xrightarrow{\eta} & TKB \\ {\scriptstyle g}\downarrow & & & & \\ R_i TC \times S_i & & & & \\ {\scriptstyle in_i}\downarrow & & & & \\ TC & & & & \end{array}$$

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

$$\begin{array}{ccccc} KA & & \xrightarrow{\quad Kf \quad} & & KB \\ {\scriptstyle g}\downarrow & & & & \downarrow \\ R_iTC \times S_i & \longrightarrow & R_iTY \times S_i & \longrightarrow & TY \end{array}$$

Since KB is connected (Assumption 15), by extensivity (Assumption 6) and definition of T (Assumption 19), $KB \to TY$ factors through $R_iTY \times S_i \hookrightarrow TY$. Since $R_iTY \times S_i \to TY$ 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 $KB \to R_iTY \times S_i$ such that the following diagram commutes.

$$\begin{array}{ccc} KA & \xrightarrow{Kf} & KB \\ {\scriptstyle g}\downarrow & & \downarrow \\ R_iTC \times S_i & \longrightarrow & R_iTY \times S_i \end{array}$$

which is equivalent to the commutation of the two following diagrams

$$\begin{array}{ccc} KA & \xrightarrow{Kf} & KB \\ {\scriptstyle g_2}\downarrow & & \downarrow \\ S_i & \xrightarrow[=]{} & S_i \end{array} \qquad \begin{array}{ccc} KA & \xrightarrow{Kf} & KB \\ {\scriptstyle g_1}\downarrow & & \downarrow \\ R_iTC & \longrightarrow & R_iTY \end{array}$$

Unique existence of $KB \to S_i$ making the left diagram commutes is ensured by the fact that S_i is orthogonal to all morphisms (Assumption 19). So a cocone on Y is essentially given by g_1 making the right diagram commute. Since R_i has a left adjoint L_i such that $L_iK = \int^j KL'_{i,j}$ (Assumption 19), the right diagram is equivalent to making the following diagram commute.

$$\begin{array}{ccc} \coprod_j KL'_{i,j}A & \xrightarrow{\coprod_j KL'_{i,j}f} & \coprod_j KL'_{i,j}B \\ {\scriptstyle g_1^*}\downarrow & & \downarrow{\scriptstyle u} \\ TC & \xrightarrow[\sigma]{} & TY \end{array}$$

Thus, this justifies the following rule

$$\frac{C \vdash (L'_{i,j}f)_j := g_1^* \Rightarrow u; \sigma \dashv Y}{C \vdash f := in_i \circ g \Rightarrow in_i \circ u^*; \sigma \dashv Y}$$

6 Coequalising phase

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

$$\coprod_i KA_i \overset{[t_i]}{\underset{[u_i]}{\rightrightarrows}} \Gamma \xrightarrow{\sigma} \Delta$$

where $t_i, u_i \in \hom_{Kl^*_T}(KB_i, \Gamma)$. We denote such a situation by

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

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 39, 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)
- $in_i : R_i TC \times S_i \hookrightarrow TC$ (operations)

In the next subsections, we discuss the different cases.

- $\eta = \dots$, or $M(\vec{x}) = ...$, in case of a successful occur-check, in Section 6.1;
- $\eta = \eta$, or $M(\vec{x}) = N(\vec{y})$, in Section 6.2, which is partly redundant with the previous case

- $in_i = in_i$, or $o(\vec{t}) = o(\vec{u})$, in Section 6.3;
- $in_I = in_I$, or $x = y$, in Section 6.4.
-

Let us mention schematically some other cases, which can never be unified in Kl_T, in most case by extensivity (Assumption 6), 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 7), i.e., when M appears deep in u.

6.1 Successful occur-check

WIP

The occur-check phase is described in more details in Section 7. Here, we assume that

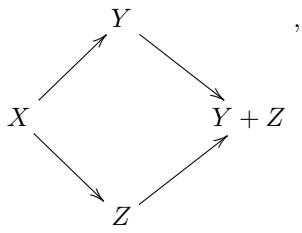
1. $C \simeq KB + C'$ for some B, C';
2. $t : KA \to TC$ factors as $KA \xrightarrow{f} KB \xhookrightarrow{in_B} TC \simeq KB + \dots$;
3. $u : KA \to TC$ factors as $KA \xrightarrow{g} TC' \xhookrightarrow{in_{TC'}} TC$.

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

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

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



,

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 $Y \leftarrow X \rightarrow Z$ 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

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

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

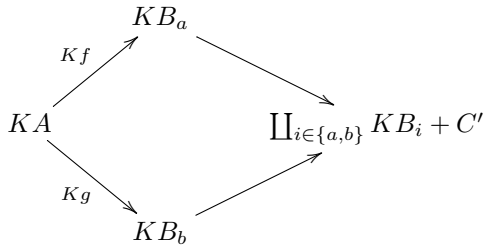
Here we are in the situation where $t, u : KA \rightarrow TC$ factor as $t', u' : KA \rightarrow C$ through $\eta : C \rightarrow TC$. Note that since postcomposition with η is precisely the left adjoint (and thus cocontinuous) functor from $\mathcal{C}$ to Kl_T, it is enough to compute the coequaliser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

Note that if a and b are distinct, then the case we describe is redundant with some specific case of the pruning phase (Section 5.2).

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

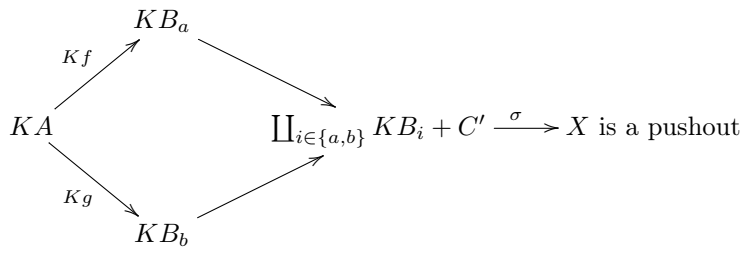


Lemma 29. *The coequaliser of a diagram of this shape, in any category which has the involved colimits, is $X + C'$, where X is the coequaliser or Kf and Kg if $a = b$, or the pushout of Kf and Kg, otherwise.*

Thanks to Assumption 11, we know that such a colimit exists (but do we need that this limit lives in D?).

All the arguments in this section justify the following rule, whose premise

can always be satisfied.



$$\coprod_{i\in\{a,b\}} KB_i + C' \vdash \eta \circ in_a \circ Kf = \eta \circ in_b \circ Kf \Rightarrow \eta \circ \sigma \dashv X$$

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

Here we assume that $t, u : KA \to TC$ factors as $t', u' : KA \to R_iTC \times S_i$ through $R_iTC \times S_i \hookrightarrow TC$. A cocone in Kl_T is given by an object Y with a morphism $C \xrightarrow{\sigma} TY$ such that the following diagram commutes.

$$\begin{array}{ccccc} KA & \xrightarrow{t',!} & R_iTC \times S_i & \xrightarrow{R_i\sigma^* \times !} & R_iTY \times S_i \\ {\scriptstyle u',!}\downarrow & & & & \downarrow{\scriptstyle in_i} \\ R_iTC \times S_i & \xrightarrow[R_i\sigma^* \times !]{} & R_iTY \times S_i & \xrightarrow[in_i]{} & TY \end{array}$$

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

$$\begin{array}{ccc} KA & \xrightarrow{t',!} & R_iTC \times S_i \\ {\scriptstyle u',!}\downarrow & & \downarrow{\scriptstyle R_i\sigma^* \times S_i} \\ R_iTC \times S_i & \xrightarrow[R_i\sigma^* \times S_i]{} & R_iTY \times S_i \end{array}$$

hich is equivalent to the commutation of the two following diagrams

$$\begin{array}{ccc} KA & \xrightarrow{!} & S_i \\ {\scriptstyle !}\downarrow & & \downarrow{\scriptstyle !} \\ S_i & \xrightarrow[!]{} & S_i \end{array} \qquad \begin{array}{ccc} KA & \xrightarrow{t'} & R_iTC \\ {\scriptstyle u'}\downarrow & & \downarrow{\scriptstyle R_i\sigma^*} \\ R_iTC & \xrightarrow[R_i\sigma^*]{} & R_iTY \end{array}$$

The left one is trivial Don't we need subterminality of S_i here? Maybe not. Since R_i has a left adjoint L_i such that $L_iK = \int^j KL'_{i,j}$ (Assumption 19), the

right one is equivalent to making the following diagram commute.

$$\begin{array}{ccc} \coprod_j KL'_{i,j}A & \xrightarrow{t'^*} & TC \\ {\scriptstyle u'^*}\downarrow & & \downarrow{\scriptstyle \sigma^*} \\ TC & \xrightarrow[\sigma^*]{} & TY \end{array}$$

Thus, this justifies the following rule

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

6.4 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.

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

7 Occur-check

WIP TODO update with the new definition

The occur-check allows to jump from the coequalising phase (Section 6) to the pruning phase (Section 5), 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 $\coprod_i JS_i \times \prod_j P_{L_{i,j}-}$. Again, JS_i is subterminal. Then, GJ is isomorphic to JF, i.e., the following square commutes up to isomorphism.

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{F} & \mathcal{C} \\ {\scriptstyle J}\downarrow & & \downarrow{\scriptstyle J} \\ \hat{D} & \xrightarrow[G]{} & \hat{D} \end{array}$$

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.

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{T} & \mathcal{C} \\ {\scriptstyle J}\downarrow & & \downarrow{\scriptstyle J} \\ \hat{D} & \xrightarrow[G^*]{} & \hat{D} \end{array}$$

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 $\hom_{\mathcal{C}}(K-, TC) + 1$. .

7.1 Preliminar induction lemma

TODO: adapt to take into account the subterminal object

Let us denote G^* by M. We want to prove by induction that the following square is a pullback for any $u : ya \to M1$

$$\begin{array}{ccc} ya \times_{T1} ya + ya & \xrightarrow{[-,u]} & M(ya+1) \\ \downarrow & & \downarrow{\scriptstyle -[u,*]} \\ ya & \xrightarrow[u]{} & M1 \end{array}$$

^In other words, we show, by induction on $u \in M1_a$, the following property: for any $f : a \to b$ and $t \in M(yd+1)_b$ such that $f \cdot u = t[u]$, then

- either $t = \eta(g)$ for some $g : a \to b$ s.t. $f \cdot u = g \cdot u$,
- either $t = wk(f \cdot u)$

Proof. We do the induction step. Assume $u = o(u_1, \dots u_n)$ with $u_i \in M1_{L_i a}$ satisfying the property above.

Let $f : a \to b$ and $t \in M(ya+1)_b$ such that $f \cdot u = t[u]$. Then, we may be in the first case. Otherwise, it must be that $t = o(t_1, \dots t_n)$ with $t_i \in M(ya+1)_{L_i b}$, and such that $L_i f \cdot u_i = t_i[o(u_1, \dots, u_n)] \in M1_{L_i b}$. Let us note that the right hand side is equal to $t_i[\delta, *][u_i, *]$, where $u_i : yL_i a \to M1$, so that $\delta : ya \to M(yL_i a + 1)$ is defined (through the yoneda lemma) as the element $o(\dots, u_{i-1}, \eta(id), u_{i+1}) \in M(yL_i a+1)_a$ with $\eta(id) \in M(yL_i a)_{L_i a}$. Note that $t'_i =^{def} t_i[\delta, *] \in M(yL_i a + 1)_{L_i b}$. We apply the induction hypothesis for $t'_i[u_i] = L_i f \cdot u_i$. For any i,

1. either $t'_i = \eta(g_i)$ for some $g_i : L_i a \to L_i b$ such that $L_i f \cdot u_i = g_i \cdot u_i$
2. either $t'_i = L_i f \cdot u_i \in M1_{L_i b}$.

The first case is in fact impossible, for in this case, we have a commuting square

$$\begin{array}{ccc} yL_ib & \xrightarrow{t_i} & M(ya+1) \\ {\scriptstyle g_i}\downarrow & & \downarrow{\scriptstyle -[\delta,*]} \\ yL_ia & \longrightarrow & M(yL_ia+1) \end{array}$$

which can be rewritten as

$$\begin{array}{ccc} yL_ib & \xrightarrow{t_i} & ya+1+GM(ya+1) \\ {\scriptstyle g_i}\downarrow & & \downarrow{\scriptstyle -[\delta,*]} \\ yL_ia & \xrightarrow[in_1]{} & yL_ia+1+GM(yL_ia+1) \end{array}$$

By extensivity, t_i cannot factor neither through $1 \to ya+1+GM(ya+1)$ nor through $GM(ya+1)$. Thus, t_i factors through $ya \to ya+1+GM(ya+1)$. But $ya \xrightarrow{\delta} M(yL_ia+1)$ factors through $GM(yL_ia+1) \to M(yL_ia+1)$ so commutation is impossible.

The second case implies that t_i is in fact in $M(A)_{L_id}$.

Indeed, then we have a commuting square

$$\begin{array}{ccc} yL_ib & \xrightarrow{t_i} & M(ya+1) \\ {\scriptstyle L_if}\downarrow & & \downarrow{\scriptstyle \delta} \\ yL_ia & \xrightarrow[u_i]{} M1 \xrightarrow[Min_2]{} & M(yL_ia+1) \end{array}$$

Let us detail the second square. Since M preserves pullbacks, it is enough to prove that the following square is a pullback.

$$\begin{array}{ccc} MB' & \longrightarrow & A+M(A+B') \\ {\scriptstyle =}\downarrow & & \downarrow \\ MB' & \longrightarrow & M(A+B') \end{array}$$

But then, it is enough to see that the following squares are pullbacks

$$\begin{array}{ccc} M\quad \emptyset & \longrightarrow & A \\ \downarrow & & \downarrow \\ \emptyset & \longrightarrow & MA \cong A+GMA \\ \downarrow & & \downarrow \\ MB' & \longrightarrow & M(A+B') \end{array}$$

Then, by Lemma 40, this squares decomposes as

$$\begin{array}{ccccc} yL_ib & \longrightarrow & M1 & \xrightarrow{Min_2} & M(ya+1) \\ {\scriptstyle L_if}\downarrow & & \downarrow{\scriptstyle =} & & \downarrow{\scriptstyle \delta} \\ yL_ia & \xrightarrow[u_i]{} & M1 & \xrightarrow[Min_2]{} & M(yL_ia+1) \end{array}$$

This means that $t_i = wk(L_if \cdot u_i)$.
Then, can we conclude anything? I think so □

Question 30. *Is there a better categorical proof? (maybe using that fact that yd is tiny)*

Corollary 31. *Given any $A \xrightarrow{\sigma} M1$, and a commuting square*

$$\begin{array}{ccc} P & \longrightarrow & M(yd+A) \\ \downarrow & & \downarrow{\scriptstyle [u,\sigma]^*} \\ yd & \xrightarrow[u]{} & M1 \end{array}$$

the top morphism factors either by $yd \to M(yd+A)$ or by $MA \to M(yd+A)$

Proof. Note that by the previous result, this is true if $A = 1$ and $\sigma = \eta_1 : 1 \to M1$. Now, the right morphism is equal to $M(yd+A) \to M(yd+1) \to M1$, so that the previous result, this means that this square decomposes as

$$\begin{array}{ccc} P & \longrightarrow & M(yd+A) \\ \downarrow & & \downarrow \\ P' & \longrightarrow & M(yd+1) \\ \downarrow & & \downarrow \\ yd & \xrightarrow[u]{} & M1 \end{array}$$

where the bottom square is a pullback.

If the middle horizontal morphism factors through $yd \to M(yd+1)$ then top one factors through $yd \to Myd$ because the following square is a pullback, by Lemma 40.

$$\begin{array}{ccc} yd & \longrightarrow & M(yd+A) \\ \downarrow & & \downarrow \\ yd & \longrightarrow & M(yd+1) \end{array}$$

Otherwise, the middle horizontal factors as $M1 \to M(yd+1)$. But then, the top one factors through $MA \to M(yd+A)$ because the following square is

a pullback, by Lemma 41

$$\begin{array}{ccc} MA & \longrightarrow & M(yd + A) \\ \downarrow & & \downarrow \\ M1 & \longrightarrow & M(yd + 1) \end{array}$$

□

Corollary 32. *Given $\sigma : \Gamma \to T1$, $u : Kd \to T1$, $f : Kc \to Kd$, if the following square commutes, then the top morphism factors through either $T\Gamma \to T(Kd+\Gamma)$ or $Kd \to TKd$*

$$\begin{array}{ccc} Kc & \longrightarrow & T(Kd + \Gamma) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle [u,\sigma]^*} \\ Kd & \xrightarrow[u]{} & T1 \end{array}$$

Proof. Apply J to this diagram, and you get

$$\begin{array}{ccc} yc & \longrightarrow & M(yd + J\Gamma) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle [u,\sigma]^*} \\ yd & \xrightarrow[u]{} & M1 \end{array}$$

then apply the previous corollary. Note that we are not working with discrete presheave here. □

Question 33. *Can we do the whole algorithm by translating it in the presheaf world?*

8 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.

9 Applications

9.1 Nominal sets (untyped)

9.2 Nominal sets, simply-typed

Let T be the set of simple types. Here we consider the category of functors $(Set_f^T \times T)_m \to Set$, where Set_f^T is the category of finite families indexed by T

(it is equivalent to the comma category J/T, where $J : \mathbb{F} \to Set$ is the canonical inclusion), and the indice $-_m$ means that we restrict to monomorphisms. Based on the paragraph before Section 8 in [3] (the author says he will present more explictly these results later, but I couldn't find a reference), we can argue that it is a Grothendieck topos based on the following fact: products of Grothendieck topos are topos.

- this category is boolean (meaning any subobject has a complement),
- has effective unions (not sure what it means)
- has finite limits (it is even complete)
- has finite coproducts (coproducts commute with pullbacks)
- there is no infinite chain of proper subobjects

Remark 34. TODO: voir si ca marche avec le cas Grothendieck.

Again, we restrict to the full subcategory of functors preserving pullbacks. As in the case presented above, representable presheaves belong to this subcategory, which is stable under limits and coproducts. It is reflective subcategory of the total category of functors (as it is the category of models of some limit-sketches) and thus to show that it is a Grothendieck topos, it is enough to show that the left adjoint preserves finite limits.

Example 35. Consider the functor for λ-calculus:

$$\begin{aligned} F(X)_{\Gamma\vdash\sigma} &= y_{\sigma\vdash\sigma}(\Gamma \vdash \sigma) && ((\text{variables})) \\ &+ \coprod_{\tau',\sigma'} X(\Gamma, \tau' \vdash \sigma') \times y_{\vdash\tau'\Rightarrow\sigma'}(\Gamma \vdash \sigma) && ((\text{abstraction})) \\ &+ \coprod_{\tau'} X(\Gamma \vdash \tau' \Rightarrow \sigma) \times X(\Gamma \vdash \tau') && ((\text{application})) \end{aligned}$$

. In other words,

$$\begin{aligned} F(X) = &\coprod_{\sigma} y_{\sigma\vdash\sigma} \\ &+ \coprod_{\tau',\sigma'} X(-, \tau' \vdash \sigma') \times y_{\vdash\tau'\Rightarrow\sigma'} \\ &+ \coprod_{\tau',\sigma} X(- \vdash \tau' \Rightarrow \sigma) \times X(- \vdash \tau') \times y_{\vdash\sigma} \end{aligned}$$

Does it satisfy the conditions? no, because the second line does not preserve the terminal object... More precisely: $X(-, \tau' \vdash \sigma') \times y_{\vdash\tau'\Rightarrow\sigma'}$ is not continuous in X.

9.3 Linear syntax

Should work with the new condition on S_i

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.

Note that $F(X)$ is of the shape $I + \coprod_i X_{p_{i,1}} \times \cdots \times X_{p_{i,m_i}} \times yn_i$ and each $X \mapsto X_{p_{i,1}} \times \cdots \times X_{p_{i,m_i}} \times yn_i$ is left adjoint to $X \mapsto X_{n_i} \times (yp_{i,1} + \cdots + yp_{i,m_i})$. No! But almost, i.e., if there exists a morphism $A \to X_p \times y_n$, then in fact $A = A_n yn$ and there exists a morphism $A_n \times yp \to X$, but the converse is false.

Remark 36. We could have done the non-linear version in this setting as well, but the abstract syntax is more convoluted (see the binomial coefficient) and metavariables must still be linear.

Example 37. linear lambda calculus, quantum lambda calculus

9.4 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 nominal set with Sn the set of types taking free type variables in X.

Now, contexts are of the shape $n; \sigma_1, \ldots, \sigma_p \vdash \tau$, where $\sigma_i, \tau \in Sn$. Categorically,

We first consider the category of functors $\int^{n \in \mathbb{F}_m} Set_f^{Sn} \times Tn \to Set$ (this is the oplax colimit), i.e., a morphism between $n, \Gamma \vdash \tau$ and $n', \Gamma' \vdash \tau'$ is a monomorphism $\sigma : n \to n'$ and renamings $\Gamma[\sigma] \to \Gamma'$ such that $\tau[\sigma] = \tau'$. This does not seem to fit in the previous framework. Indeed, what are the connected components? Do corpoducts exist in each of these components? . Are they boolean? So instead, we could take $\coprod_{n \in \mathbb{F}_m}$ rather than the oplax colimit. But then the metavariables are not so nice. Can we adapt manually the proof that the cat of nominal sets is a topos (in the elephant)?

Example 38. Following [2], we specify System F with the functor

$$\begin{aligned} F(X)_{n;\Gamma\vdash\sigma} &= V_{n;\Gamma\vdash\sigma} && \text{(variables)} \\ &+ \coprod_{\tau} X(n+1;\Gamma\vdash\tau)\delta_{\sigma,\forall\tau} && \text{(type abstraction)} \\ &+ \coprod_{\tau'} X(n;\Gamma\vdash\forall\tau')\times(\sigma':Sn)\delta_{\sigma,\tau'[\sigma']} && \text{(type application)} \\ &+ \coprod_{\tau} X(n;\Gamma,\tau\vdash\sigma')\delta_{\sigma,\tau\Rightarrow\sigma'} && \text{(abstraction)} \\ &+ \coprod_{\tau'} X(n;\Gamma\vdash\tau'\Rightarrow\sigma)\times X(n;\Gamma\vdash\tau') && \text{(application)} \end{aligned}$$

Here, V is defined as $V(n;\Gamma\vdash\sigma) = \#\{\sigma\in\Gamma\}$.

10 Conclusion and future work

It would be nice to allow metavariables with mixed linearity constraints.

References

[1] Menni MatÃas Fiore, Marcelo. Reflective kleisli subcategories of the category of eilenberg-moore algebras for factorization monads. *Theory and Applications of Categories [electronic only]*, 15:40–65, 2005.

[2] Makoto Hamana. Polymorphic abstract syntax via grothendieck construction. 2011.

[3] Matías Menni. About n-quantifiers. *Applied Categorical Structures*, 11(5):421–445, 2003.

A Some lemmas

Lemma 39. *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 $A_2 \rightrightarrows B \to C$.*

Lemma 40. *Let M be an algebraically free monad on an extensive category. Then, for any Kleisli morphism $B \to M(B')$, for any object A, the following square is a pullback (TODO: refine the assumptions)*

$$\begin{array}{ccc} A & \longrightarrow & M(A+B) \\ \downarrow & & \downarrow \\ A & \longrightarrow & M(A+B') \end{array}$$

Proof. By Lambek's lemma, $M(X) \cong X + GMX$ where $M = G^*$. Through this isomorphism, this square becomes

$$\begin{array}{ccc} A & \longrightarrow & A+B+GM(A+B) \\ \downarrow & & \downarrow \\ A & \longrightarrow & A+B'+GM(A+B') \end{array}$$

where horizontal morphisms are left coproduct injections. TODO: conclude by extensivity. □

Lemma 41. *Let M be an algebraically free cartesian monad on an extensive category. Then, for any Kleisli morphism $B \to MB'$, the following square is a pullback (TODO: refine the assumptions)*

$$\begin{array}{ccc} MB & \longrightarrow & M(A+B) \\ \downarrow & & \downarrow \\ MB' & \longrightarrow & M(A+B') \end{array}$$

Proof. This commuting square expands as

$$\begin{array}{ccc} MB & \xrightarrow{Min_2} & M(A+B) \\ \downarrow & & \downarrow \\ MMB' & \longrightarrow & M(A+M(A+B')) \\ {\scriptstyle =}\downarrow & & \downarrow \\ MMB' & \xrightarrow{MMin_2} & MM(A+B') \\ {\scriptstyle \mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ MB' & \xrightarrow[Min_2]{} & M(A+B') \end{array}$$

where each square is a pullback square. □

Lemma 42. *Grothendieck topoi are stable under products.*

Proof. Let $(A_i)_{i \in I}$ be a family of Grothendieck topoi. Then, there exists a family of small categories $(B_i)_{i \in I}$ such that each A_i is reflective subcategory of $\hat{B}_i$ and moreover, the reflector is left exact. Then, $\prod_i A_i \to \prod_i \hat{B}_i \cong \widehat{\coprod_i B_i}$ is also coreflective and the reflector is left exact, concluding the proof. □