Towards Generic Semantics for Substructural Generic Quantification

Thibaut Antoine
Univ Rennes, CNRS, IRISA

David Baelde
Univ Rennes, CNRS, IRISA

Miller and Tiu's generic quantification (i.e., the ∇ quantifier) brings a logical treatment of the common concepts of name and freshness. Unlike the related fresh quantifier of nominal logic, generic quantification has been defined proof-theoretically. In fact, several generic quantifiers have been considered. In the earlier version, nabla is defined as a self-dual quantifier commuting with all propositional connectives. Later, it has been extended to satisfy two additional properties: $\nabla x.\nabla y.F$ is equivalent to $\nabla y.\nabla x.F$; $\nabla x.F$ is equivalent to F if x does not occur in F. Together, the two properties imply nominal logic's equivariance principle.

While they have not been considered independently so far, it is natural to consider the two properties individually, giving rise to four quantifiers. Taking inspiration from substructural proof theory, we call the first property *exchange* and the second one *weakening*, and define uniformly four logics through proof systems with substructural generic contexts. We then consider the problem of giving a (sound and complete) truth semantics for these logics. We build upon earlier work by Goubault-Larrecq, which proposed a complete semantics for Miller and Tiu's early nabla quantifier, and show that (enriched) nabla sets can be used to define, in a uniform way, a semantics for the four substructural generic logics. We show that this semantics is sound and complete for three of the logics, defined through their natural (classical) sequent calculi. However, we identify a mismatch for the case where nabla satisfies weakening but not exchange: in that case, the natural proof system is not sound for our semantics.

1 Introduction

It is not uncommon in computer science and mathematics to reason with objects that are generic. A prominent example is capture-avoiding substitution, which can be implemented by α-renaming bound variables on the fly, i.e. replacing the bound variable by a *fresh* one. The fresh variable is generic in that context: it does not satisfy any particular property besides being a variable. Similarly, when proving $\forall x.F$ in natural deduction, one picks an arbitrary, fresh *eigenvariable* c to prove $F[x := c]$. There are many other examples involving e.g. memory addresses in programming, node identifiers in XML, or independent random samplings such as the nonces of cryptography. The general framework of nominal sets [8] can be used to formally represent and reason about these varied applications. Nominal sets feature specific objects called names which can be exchanged; all operations commute with name exchange, and the equivariance principle imposes that all objects (and statements) are considered up to name exchange. Nominal sets have served as a foundation for formal languages with data, e.g. with nominal automata [3]. More generally, they are the foundation for nominal logics [4, 7, 11] where the primitive notions of name, exchange and support serve to derive the signature quantifier N which serves to introduce fresh names.

Submitted to:
LFMTP 2026

A proof-theoretic alternative to the nominal logics initiated by Pitts and Gabbay is the line of work initiated by Miller and Tiu with the introduction of the ∇ quantifier in [6]. In that paper, they show that a logic with generic quantification can be defined purely proof-theoretically. The ∇ quantifier of [6] is essentially defined by the fact that it is self-dual; it commutes with propositional connectives; it commutes with first-order quantifiers in a particular way, involving a technique called *raising* that is made possible by the consideration of higher-order terms. Unlike the fresh quantifier of nominal, which only makes sense over name types, generic quantification can take place at all types: in Miller and Tiu's setting, there is no difficulty e.g. with considering generic functions. Despite successful applications of that first logic, difficulties have motivated Tiu to propose the richer logic LG [10] where ∇ satisfies two additional equivalences: $\nabla x.\nabla y.F$ is equivalent to $\nabla y.\nabla x.F$; $\nabla x.F$ is equivalent to F when x does not occur free in F. These two equivalences amount to the equivariance principle of nominal sets. This second flavour of ∇ has been incorporated in the logic of the proof assistant Abella [1] which has been used successfully on various applications in the meta-theory of programming languages.

The logics featuring ∇ have all been defined proof-theoretically. This makes ∇ largely independent of the underlying logic: it has always been defined in an intuitionistic setting, but it can be naturally be reframed in classical sequent calculus or, conversely, in more restrictive substructural logics such as linear logic; there is no doubt that the cut elimination theorems justifying the logics would carry over. However, the lack of semantics for ∇ is an issue: Tarski-style semantics remain the most common way to understand the meaning of a logic, and they can be a powerful additional tool to analyze derivability. As a result, some works have been devoted to elaborating semantics for generic quantification. To the best of our knowledge, Goubault-Larrecq [5] provides the most satisfying answer to that problem. He notably improves upon previous attempts [9] by allowing generic quantification over all types, as should be the case. In that work, Goubault-Larrecq introduces *nabla sets*, which are $\mathbb{N}$-indexed chains of sets equipped with a notion of new element at each step in the chain. He proves that the classical variant of Miller and Tiu's original proof system is sound and complete with respect to his semantics (we ignore, for now, an assumption under which completeness holds). In particular, the semantics invalidates the above-mentionned equivalences on ∇ that hold in LG [10].

The work of Goubault-Larrecq leaves open several questions, which we tackle in this paper. First, his completeness result only holds under the assumption that there is a single base type. We view this as a severe limitation, since we are concurrently working on incorporating generic quantification into some logic used for cryptographic reasoning [2], where ∇ would be used to introduce independent random samplings, with different probability distributions for different base types. Second, it is natural to attempt to extend Goubault-Larrecq's semantics for the later flavour of ∇. In fact, we frame the question even more generally, and consider a lattic of four proof systems, corresponding to accepting any combination of the two equivalences that give rise to Tiu. Taking inspiration in substructural proof theory, we view $\nabla x.\nabla y.F \equiv \nabla y.\nabla x.F$ as an exchange rule and $\nabla x.F \equiv F$ as a weakening rule, and we coin our four flavours of ∇ as substructural generic quantification. We present these proof systems in section 2. We then introduce in section 3 an enriched form of nabla sets, and we show in section 4 how they can be used to provide a semantics for the four flavours of generic quantification. We note, however, that when only weakening is considered, our proof system is not sound with respect to the expected semantics, a difficulty that we analyze throughout the paper and that calls for future works discussed in conclusion. For the other three logics, soundness holds and we also show completeness in section 5. We conclude with several directions for future work in section 6.

2 Proof systems

In this section, we define four sequent calculi corresponding to classical first-order logic (over higher-order terms) with four flavors of generic quantification. The simplest system is the classical variant of Miller and Tiu's original generic quantification [6], i.e. the exact system considered by Goubault-Larrecq in [5]. More expressive systems are obtained by adding structural rules for ∇, the most expressive one corresponding to the classical variant of Tiu's logic LG [10].

We consider higher-order terms, noted t, u, v, ... which are standard simply-typed λ-terms, considered modulo $\alpha\beta\eta$-equivalence. Types will be noted using the letter τ. We then consider first-order formulas extended with a ∇ quantifier:

$$F ::= \bot \mid P(t_1, \ldots, t_n) \mid F \Rightarrow F' \mid \forall x : \tau.F \mid \nabla x : \tau.F$$

Above, P is a predicate symbol; these symbols are typed, but we do not detail the obvious typing rules. We also use the standard notions of free variables, noted $\mathrm{fv}(F)$, and capture-avoiding substitution, noted $F[x := t]$. As is standard in classical logic, we consider above a minimal set of connectives for simplicity; conjunction, disjunction and existential quantification can be expressed from these primitive connectives.

We work with sequents of the form $\Gamma \vdash \Delta$, where Γ and Δ are finite multisets of *generic judgements*. These judgements, noted using the letter J, are of the form $\sigma \rhd F$, embedding a formula F under a signature $\sigma = (x_1 : \tau_1, \ldots, x_n : \tau_n)$ which is a (possibly empty) sequence of distinct typed variable declarations. The meaning of such a judgement J is the formula $|J| = \nabla\sigma.F = \nabla x_1 : \tau_1 \ldots \nabla x_n : \tau_n.F$. Consistently with that meaning, the variables of σ are considered as bound variables in J, and judgements are considered modulo α-renaming of these variables. As usual, a sequent $\Gamma \vdash \Delta$ reads as the formula $|\Gamma \vdash \Delta| := \bigwedge_{J \in \Gamma} |J| \Rightarrow \bigvee_{J' \in \Delta} |J'|$. The remaining free variables in $|\Gamma \vdash \Delta|$ can be understood as being universally quantified.

$$\frac{}{\Gamma, \sigma \rhd \bot \vdash \Delta}\ (\bot_L) \qquad \frac{}{\Gamma, \sigma \rhd F \vdash \sigma \rhd F, \Delta}\ (\mathrm{Ax})$$

$$\frac{\Gamma \vdash \sigma \rhd F, \Delta \quad \Gamma, \sigma \rhd G \vdash \Delta}{\Gamma, \sigma \rhd (F \Rightarrow G) \vdash \Delta}\ (\Rightarrow_L) \qquad \frac{\Gamma, \sigma \rhd F \vdash \sigma \rhd G, \Delta}{\Gamma \vdash \sigma \rhd (F \Rightarrow G), \Delta}\ (\Rightarrow_R)$$

$$\frac{\Gamma, \sigma \rhd F[x := t] \vdash \Delta}{\Gamma, \sigma \rhd \forall x : \tau.F \vdash \Delta}\ (\forall_L) \qquad \frac{\Gamma \vdash \sigma \rhd F[x := h\ \sigma], \Delta \quad h : \sigma \to \tau \notin \mathrm{fv}(\Gamma, \Delta, F)}{\Gamma \vdash \sigma \rhd \forall x : \tau.F, \Delta}\ (\forall_R)$$

$$\frac{\Gamma, (\sigma, x : \tau) \rhd F \vdash \Delta}{\Gamma, \sigma \rhd \nabla x : \tau.F \vdash \Delta}\ (\nabla_L) \qquad \frac{\Gamma \vdash (\sigma, x : \tau) \rhd F, \Delta}{\Gamma \vdash \sigma \rhd \nabla x : \tau.F, \Delta}\ (\nabla_R)$$

Figure 1: Core sequent calculus rules

We show in fig. 1 the core rules of our proof systems, which are the the rules of [6] but for classical sequents. Most rules are straightforward adaptations of the usual rules of classical first-order sequent calculus to work with generic judgements. In the $\forall_L$ rule, the term t must have type τ, and can mention variables from σ. The ∇ rules simply enrich the signature of generic judgements. The most interesting rule is $\forall_R$, which makes crucial use of higher-order

terms to express scoping through *raising* [6]: when introducing a universally quantified variable $x : \tau$ under a generic signature $\sigma = (x_1 : \tau_1, \ldots, x_n : \tau_n)$, we account for the fact that x may depend on the variables in σ by replacing it with $h\ \sigma = h\ x_1 \ldots x_n$ where h is a fresh variable of type $\sigma \to \tau = \tau_1\ \to \ldots \to \tau_n \to \tau$. Implicitly, h is universally quantified at toplevel in the sequent, but as such it cannot directly depend on the variables $x_1, \ldots, x_n$, hence the explicit application. The reader may note that raising is similar in spirit to skolemization.

$$\frac{\Gamma, J, J \vdash \Delta}{\Gamma, J \vdash \Delta}\ C_L \quad \frac{\Gamma \vdash J, J, \Delta}{\Gamma \vdash J, \Delta}\ C_R \quad \frac{\Gamma \vdash \Delta}{\Gamma, J \vdash \Delta}\ W_L \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash J, \Delta}\ W_R$$

$$\frac{\Gamma, J' \vdash \Delta \quad J \approx J'}{\Gamma, J \vdash \Delta}\ (\approx_L) \qquad \frac{\Gamma \vdash J', \Delta \quad J \approx J'}{\Gamma \vdash J, \Delta}\ (\approx_R)$$

Figure 2: Core structural rules

$$\frac{x \notin \mathrm{fv}(F, \sigma, \sigma')}{(\sigma, x : \tau, \sigma') \rhd F \approx (\sigma, \sigma') \rhd F}\ \mathsf{w} \qquad \frac{}{(\sigma, x : \tau, y : \tau', \sigma') \rhd F \approx (\sigma, y : \tau', x : \tau, \sigma') \rhd F}\ \mathsf{x}$$

Figure 3: Structural rules for signatures

We also include in all of our calculi some core structural rules, shown in fig. 2: this includes the usual contraction and weakening rules of classical logic, but also a rule for replacing a generic judgement with an equivalent one. The latter notion relies on a judgement $J \approx J'$ which is defined by the core rules of reflexivity, symmetry and transitivity (not shown here) and, optionally, some rules among the generic weakening[1] and exchange rules shown in fig. 3. For each of the four combinations $X \subseteq \{\mathsf{w}, \mathsf{x}\}$ of the two structural rules for generic contexts, we obtain a logic that we call $\mathrm{FO}\lambda^{\nabla}_X$. For conciseness, we write X as a word, e.g. $\mathrm{FO}\lambda^{\nabla}_{\mathsf{wx}}$ is $\mathrm{FO}\lambda^{\nabla}_{\{\mathsf{w},\mathsf{x}\}}$.

As already mentionned, $\mathrm{FO}\lambda^{\nabla}$ is obviously the logic considered in [5], i.e. the classical variant of Miller and Tiu's $\mathrm{FO}\lambda^{\nabla}$ [6]. The systems $\mathrm{FO}\lambda^{\nabla}_{\mathsf{w}}$ and $\mathrm{FO}\lambda^{\nabla}_{\mathsf{x}}$ have never been considered as far as we know, though they are natural. Finally, we view $\mathrm{FO}\lambda^{\nabla}_{\mathsf{wx}}$ as the classical variant of Tiu's LG [10]: the two systems differ, but only in ways that should not affect provability. Indeed, LG does not rely on generic signatures, using instead nominal constants to represent generically quantified variables. Hence a formula F in an LG sequent can be viewed as the generic judgement $a_1, \ldots, a_n \rhd F$ for any collection of names $a_1, \ldots, a_n$ containing all names occurring in F. Because the implicit generic signature of LG can virtually grow as needed, there is no need for our w rule. Finally, the axiom in LG proves $F \vdash G$ when the two formulas are equivariant, i.e. equal up to a renaming, which internalizes the x rule.

Although we are concerned in this paper with giving sound and complete semantics for our four calculi, we conclude this section with a few simple proof theoretic observations. As usual, the initial rule (Ax) can be reduced to its atomic form: the general (Ax) rule is indeed derivable

[1] Weakening is poor terminology, since w is as much a strengthening as a weakening. Calling this principle *support independence* might be preferable.

in the calculus where (Ax) is given only on atomic formulas. However, the same does not hold for the $(\approx_L)$ and $(\approx_R)$ rules when rule w is allowed. Said otherwise, the $(\approx)$ steps cannot be absorbed into the (Ax) rule, as one might expect from a naive analogy with LG. The problematic case is the following (we do not show types, which are irrelevant here):

$$\dfrac{\dfrac{\Gamma \vdash \sigma, y \rhd F[x := h\ \sigma\ y], \Delta}{\Gamma \vdash \sigma, y \rhd \forall x.F, \Delta}\ (\forall_R)}{\Gamma \vdash \sigma \rhd \forall x.F, \Delta}\ (\approx_R)$$

Here, the use of $(\approx_R)$ with w justifying the elided premise introduces an extra generic variable y, which is then taken into account when introducing x. In that case, it is not possible to permute the $(\approx_R)$ rule above the $(\forall_R)$. Beyond this failure of permutation-based elimination, the semantics of the next sections can be used to show that restricting $(\approx_R)$ does impact provability.

The above difficulty is not an issue in itself, but a similar difficulty casts doubts regarding the possibility of cut elimination in our logics. A typical cut that is difficult to reduce is shown next, where we use $(\approx)$ and w in the right subderivation:

$$\dfrac{\dfrac{\Gamma \vdash \sigma \rhd F[x := h\ \sigma], \Delta}{\Gamma \vdash \sigma \rhd \forall x.F, \Delta}\ (\forall_R) \qquad \dfrac{\dfrac{\Gamma, \sigma, y \rhd F[x := t] \vdash \Delta}{\Gamma, \sigma, y \rhd \forall x.F \vdash \Delta}\ (\forall_L)}{\Gamma, \sigma \rhd \forall x.F \vdash \Delta}\ (\approx_L)}{\Gamma \vdash \Delta}\ (cut)$$

Here, the term t can contain the variable y, but h in the left subderivation can only be substituted for a term that does not feature generic variables. This problem arises already in LG, and Tiu uses special transformations rules to deal with it [10, Lemma 3.4 and 3.5], but we conjecture that these cannot be adapted for the cases where w is allowed but not x. In any case, our completeness results show that the cut rule is admissible for all logics but $\mathrm{FO}\lambda^{\nabla}_{\mathsf{w}}$, but this does not yield an effective syntactic cut elimination procedure.

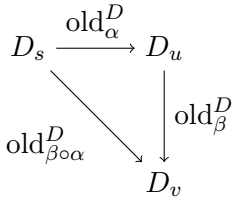
3 Enriched nabla sets

In [5], Goubault-Larrecq introduces *nabla sets* to provide a semantics for $\mathrm{FO}\lambda^{\nabla}$, which he shows to be complete when there is a single base type (though generic quantification can still be used over any of the arrow types formed from this single base type). We enrich nabla sets to provide a more flexible framework for generic quantification over arbitrary types, as defined next.

Roughly, Goubault-Larrecq's nabla sets are collections of domains D_i indexed by natural numbers, with injections from each D_i to the next one D_{i+1}. We generalize this idea by taking domains D_s for all words s over an arbitrary alphabet, equipped with injections along each word embedding. If $u, v \in \Sigma^*$ are finite words over the (potentially infinite) alphabet Σ, we write $f : u \hookrightarrow v$ and say that f is a morphism[2] from u to v when f is an injection from $\{1, \dots, |u|\}$ to $\{1, \dots, |v|\}$ such that $v_{f(i)} = u_i$ for all $1 \leq i \leq |u|$. When S is a set, the identity function over S is noted id_S. For a word $s \in \Sigma^*$, we note id_s the identity injection from s to itself: $\mathrm{id}_s = \mathrm{id}_{\{1,\dots,|s|\}}$.

Definition 3.1. A *nabla-set on* Σ is a collection of sets $D = (D_s)_{s \in \Sigma^*}$ together with a collection of injective functions $\mathrm{old}^D_\alpha : D_s \hookrightarrow D_u$ for each $s, u \in \Sigma^*$ and $\alpha : s \hookrightarrow u$, such that (1) $\mathrm{old}^D_{\mathrm{id}_s} = \mathrm{id}_{D_s}$ for every s; and (2) $\mathrm{old}^D_\beta \circ \mathrm{old}^D_\alpha = \mathrm{old}^D_{\beta \circ \alpha}$ for every s, u, v and $\alpha : s \hookrightarrow u$, $\beta : u \hookrightarrow v$:

[2] We are implicitly considering here the category whose objects are words and morphisms are word embeddings.



Intuitively, a nabla set gives a domain D_s for any s, which should be seen as an abstract generic signature. Further, an element in D_s can always be seen as an element in D_u when $s \hookrightarrow u$, modulo the corresponding old map. From the logical point of view, this is necessary because a term occurring under a generic signature σ should still make sense when we modify the signature by introducing a nabla or performing an $(\approx)$ rule.

We simply write $\mathrm{old}_{u\hookrightarrow v}$ when there is a canonical injection $\alpha : u \hookrightarrow v$, in which case it stands for old_α. For example, $\mathrm{old}_{u\hookrightarrow uv}$ is old_α where α is the injection that maps u to the prefix of uv, i.e. the identity function over $\{1,\dots,|u|\}$.

Example 3.2. We illustrate enriched nabla-set using *nominal* λ-terms, which are simply-typed λ-terms that can contain distinguished constants that we call *nominals*. For instance, if a is a nominal of type $\tau \to \tau'$ and x is a variable of type τ, then $(a\ x)$ is a nominal λ-term of type τ'.

Let Σ be the set of simple types, and fix an arbitrary type τ. For $s = \tau_1 \dots \tau_n \in \Sigma^*$ we define D_s as the set of nominal λ-terms of type τ featuring nominals among $a_1,\dots,a_n$ with a_i having type τ_i. Then, for $\alpha : u \hookrightarrow v$ and $t \in D_u$, we define $\mathrm{old}_\alpha(t) \in D_v$ as the term t where each nominal constant a_i has been replaced by $a_{\alpha(i)}$. This makes sense because the type u_i of a_i in D_u is equal to the type $v_{\alpha(i)}$ of $a_{\alpha(i)}$ in D_v.

Remark 3.3. Integers are isomorphic to words over a singleton alphabet, but our generalization goes beyond this: our enriched nabla sets have more structure than Goubault-Larrecq's nabla sets even when the alphabet is a singleton $\{a\}$, because we consider all injections. For instance, Goubault-Larrecq's nabla sets have a single old map from D_a to D_{aa}, while we have two. Further, we have two old functions from D_{aa} to itself, while Goubault-Larrecq's considers none. This extra structure is needed to account for the logics $\mathrm{FO}\lambda^\nabla_X$ for $X \neq \emptyset$.

3.1 Nabla maps and relations

We will consider maps between (enriched) nabla sets, with constraints on how they interact with (classes) of old maps. We first define the relevant classes of morphisms, corresponding to possible subsets $X \subseteq \{\mathsf{w},\mathsf{x}\}$ which we write, again, as words:

- $\alpha : s \overset{\mathsf{wx}}{\hookrightarrow} u$ denotes any word morphism from s to u, it is the same as $\alpha : u \hookrightarrow v$;
- $\alpha : s \overset{\mathsf{w}}{\hookrightarrow} u$ denotes an order-preserving morphism from s to u, i.e. $\alpha(i) \leq \alpha(j)$ when $i \leq j$;
- $\alpha : s \overset{\mathsf{x}}{\hookrightarrow} u$ denotes a permutation between s and u;
- $\alpha : s \overset{\varnothing}{\hookrightarrow} u$ denote the identity morphism from s to $u = s$.

Obviously, we have that $\alpha : u \overset{X}{\hookrightarrow} v$ and $X \subseteq Y$ imply $\alpha : u \overset{Y}{\hookrightarrow} v$.

Definition 3.4. Let D, E be nabla sets on Σ and $f = (f_s : D_s \to E_s)_{s\in\Sigma^*}$ be a collection of maps. Let $X \subseteq \{\mathsf{w},\mathsf{x}\}$. We say that f is an X-nabla map when, for every pair of words s, u and $\alpha : s \overset{X}{\hookrightarrow} u$, we have $f_u \circ \mathrm{old}^D_\alpha = \mathrm{old}^E_\alpha \circ f_s$:

$$\begin{array}{ccc} D_s & \xrightarrow{f_s} & E_s \\ {\scriptstyle \mathrm{old}^D_\alpha}\big\downarrow & & \big\downarrow{\scriptstyle \mathrm{old}^E_\alpha} \\ D_u & \xrightarrow{f_u} & E_u \end{array}$$

In that case, we write $f : D \xrightarrow{X} E$.

Note that every family of functions is a $\varnothing$-nabla map. Moreover, the composition of two X-nabla maps is also an X-nabla map.

Definition 3.5. Nabla sets and families of functions (i.e. $\varnothing$-nabla maps) form a category that we denote $\boldsymbol{\nabla}$. Considering X-nabla maps instead of $\varnothing$-nabla maps gives rise to a sub-category (obviously not full) of $\boldsymbol{\nabla}$ that we denote $\boldsymbol{\nabla}^X$.

Remark 3.6. Each $\boldsymbol{\nabla}^X$ category has all products. In particular, there is an terminal object $\mathsf{1}$ given by $\mathsf{1}_s = \{*\}$ and $\mathrm{old}^{\mathsf{1}}_\alpha = \mathrm{id}_{\{*\}}$. Products are defined component-wise: if $(D[i])_{i\in I}$ is a family of nabla sets, the product $D := \prod_{i\in I} D[i]$ is defined as follows: $D_s := \prod_{i\in I} (D[i])_s$, and if $\alpha : s \hookrightarrow u$, then $\mathrm{old}^D_\alpha := (x_i)_{i\in I} \mapsto (\mathrm{old}^{D[i]}_\alpha(x_i))_{i\in I}$.

Goubault-Larrecq interprets a formula as a $\varnothing$-nabla map from the nabla set representing the value of its free variables to the nabla set $\mathsf{2}$ of booleans, defined by $\mathsf{2}_s = \{0,1\}$ for any s and $\mathrm{old}^{\mathsf{2}}_\alpha = \mathrm{id}_{\{0,1\}}$ for every α. An X-nabla map from D to $\mathsf{2}$ will simply be called an X-nabla relation. Depending on the logic, we will consider X-nabla relations for all possible $X \subseteq \{\mathsf{w},\mathsf{x}\}$.

3.2 Pointed nabla set and nabla universes

We define the remaining ingredients needed to interpret our logics. First, we need a way to interpret the newly introduced generic variables, i.e. an analogue of the new elements of Goubault-Larrecq. In our enriched setting, this becomes a bit more complicated, because a nabla set corresponds to terms of a fixed type τ (as in example 3.2) while the type of new elements needed in D_s will correspond to the letters of s. To allow this, we introduce a distinguished subset Σ_D in our notion of pointed nabla set.

Definition 3.7. A *pointed nabla set on* Σ is a nabla set D on Σ equipped with a subset $\Sigma_D \subseteq \Sigma$ and, for each $a \in \Sigma_D$ and $s \in \Sigma^*$, a special element $\mathrm{new}^D_{sa} \in D_{sa}$ such that for every $\alpha : sa \hookrightarrow uav$:

$$\mathrm{old}^D_\alpha(\mathrm{new}^D_{sa}) = \mathrm{old}^D_{ua\hookrightarrow uav}(\mathrm{new}^D_{ua})$$

When we write $\alpha : sa \hookrightarrow uav$ above, we implicitly decompose the target word according to where α sends the final a in sa. More formally, we require that $\alpha(|sa|) = |ua|$, i.e. α maps the index of the obvious occurrence of a in sa to the index of the obvious occurrence of a in uav.

Remark 3.8. Pointed nabla sets on Σ together with X-nabla maps form a category — X-nabla maps do not have to interact in a particular way with respect to new elements. Contrary to the $\boldsymbol{\nabla}^X$ category, it does not have all products (in particular, there is no terminal object).

We can now introduce the final ingredients needed for our semantics; the next notions are adapted straightforwardly from [5]. In the rest of this section, we let Σ be the set of simple types. For convenience, we adopt an intrinsic typing style, i.e. we consider a set of variables $\mathcal{X}$ where each variable comes with a fixed type. We write x_τ to denote a variable of type τ.

Definition 3.9. A *nabla universe* S is given by the following:

- For each type τ, a pointed enriched nabla-set $S[\![\tau]\!]$ over Σ. We define the set Env of environments as the product $\prod_{x_\tau \in \mathcal{X}} S[\![\tau]\!]$. The set Env_s is the set of semantic assignments at level s, that is the set of applications σ such that $\sigma(x_\tau) \in S[\![\tau]\!]_s$ for all $x_\tau \in \mathcal{X}$.
- For each type τ, a set $S(\tau)$ of nabla-maps $\mathsf{Env} \multimap S[\![\tau]\!]$ containing the maps $\pi_{x_\tau} : \sigma \mapsto \sigma(x_\tau)$.
- For all types τ and θ, a nabla-map $\mathsf{App} : S[\![\tau \to \theta]\!] \times S[\![\tau]\!] \multimap S[\![\theta]\!]$ such that if $f \in S(\tau \to \theta)$ and $g \in S(\tau)$, then $\mathsf{App} \circ \langle f, g \rangle \in S(\theta)$.
- For all types τ and θ and $x_\theta \in \mathcal{X}$, a *function* $\Lambda_{x_\theta} : S(\tau) \to S(\theta \to \tau)$.

The above ingredients allow to interpret any λ-term $t : \tau$ as an element $S[\![t]\!] \in S(\tau)$, as follows:

$$\begin{array}{rcl} S[\![x_\tau]\!] & := & \pi_{x_\tau} \\ S[\![t_1\ t_2]\!] & := & \mathsf{App} \circ \langle S[\![t_1]\!], S[\![t_2]\!] \rangle \\ S[\![\lambda(x:\tau).t]\!] & := & \Lambda_{x_\tau}(S[\![t]\!]) \end{array}$$

For this interpretation to be correct, we require that the following natural properties hold:

- For any t and u that are $\alpha\beta\eta$-equivalent, we have $S[\![t]\!] = S[\![u]\!]$.
- For any $t : \tau$ and $y \notin \mathrm{fv}(t)$ we have, for all $s \in \Sigma^*$ and $\sigma, \sigma' \in \mathsf{Env}_s$:

$$(\forall x \neq y, \sigma(x) = \sigma'(x)) \Rightarrow S[\![t]\!]_s \sigma = S[\![t]\!]_s \sigma'$$

- For all $t : \tau, u : \theta$, $s \in \Sigma^*$ and $\sigma \in \mathsf{Env}_s$, we have $S[\![t[x_\theta := u]]\!]_s \sigma = S[\![t]\!]_s \sigma[x_\theta \mapsto S[\![u]\!]_s \sigma]$.

We finally state a key condition that nabla universes will have to satisfy in order for the $(\forall_R)$ rule to be sound.

Definition 3.10. We say that a nabla universe S has *enough maps* when for all types τ and θ, and for all $s \in \Sigma^*$ and $x \in S[\![\theta]\!]_{s\tau}$, there is an $f \in S[\![\tau \to \theta]\!]_s$ such that:

$$\mathsf{App}_{s\tau}(\mathrm{old}_{s \hookrightarrow s\tau}(f), \mathrm{new}^{S[\![\tau]\!]}_{s\tau}) = x$$

4 Semantics

We now define the semantics for our four logics. When S is a nabla universe, we write $S[\![\tau_1, \ldots, \tau_k]\!]$ for the nabla set $S[\![\tau_1]\!] \times \cdots \times S[\![\tau_k]\!]$. Note that by making the correspondence between a subset and its characteristic function, we can also see an X-nabla relation R on $\tau_1, \ldots, \tau_k$ as a nabla set R such that for every $s \in \Sigma^*$, $R_s \subseteq S[\![\tau_1, \ldots, \tau_k]\!]_s$ and, for any $\alpha : s \xhookrightarrow{X} u$ and $x \in S[\![\tau_1, \ldots, \tau_k]\!]_s$, we have $x \in R_s$ iff $\mathrm{old}_\alpha(x) \in R_u$.

Definition 4.1. An X-nabla structure S is a nabla universe S with enough maps, equipped with an X-nabla relation $S[\![P]\!]$ on $\tau_1, \ldots, \tau_k$ for each predicate symbol P of arity $\tau_1, \ldots, \tau_k$.

We now define our satisfaction function $S, \nu \models^X_s F$ where F is a formula, S is a nabla structure, $s \in \Sigma^*$ and $\nu \in \mathsf{Env}_s$ and F. We elide X when it is irrelevant or clear from the context. The novel point in this definition is the treatment of $\forall$, on which we comment immediately after.

Definition 4.2. The $\models^X$ satisfaction relation, written $\models$ for short below, is given by:

$$\begin{array}{llll} S,\nu & \not\models_s & \bot & \\ S,\nu & \models_s & P(t_1,\dots,t_k) & \iff (S[\![t_1]\!]_s\nu,\dots,S[\![t_k]\!]_s\nu) \in S[\![P]\!]_s \\ S,\nu & \models_s & F \Rightarrow G & \iff S,\nu \not\models_s F \text{ or } S,\nu \models_s G \\ S,\nu & \models_s & \nabla x:\tau.F & \iff S,\mathrm{old}_{s\hookrightarrow s\tau}(\nu)[x \mapsto \mathrm{new}^S_{s\tau}] \models_{s\tau} F \\ S,\nu & \models_s & \forall x:\tau.F & \iff S,\mathrm{old}_\alpha(\nu)[x \mapsto d] \models_u F \text{ for every } u,\alpha : s \overset{X}{\hookrightarrow} u \text{ and } d \in S[\![\tau]\!]_u \end{array}$$

In the ∇ case, we move from level s to $s\tau$, update the assignment ν using the canonical injection, and extend it to interpret x as $\mathrm{new}^S_{s\tau}$. The $\forall$ case is the only one where X is relevant. When $X = \emptyset$, the definition the only possible value for u is s itself, and our definition coincides with the one proposed by Goubault-Larrecq. That original definition works well for $X = \emptyset$ but is not adequate in other cases: in order for the $(\approx)$ rule to be sound when X is non-trivial, we need that the semantics of a formula to be preserved by X-morphisms. For instance, when $\mathsf{x} \in X$ is allowed, we need that $S,\nu \models^X_{u\tau\theta v} F$ is equivalent to $S,\nu \models^X_{u\theta\tau v} F$. We have naturally imposed this condition in definition 4.1 for atoms, and it only carries over to arbitrary formulas thanks to our semantics for $\forall$ which considers all possible extensions of the current level.

Lemma 4.3. *Let F be a formula. Let $X \subseteq \{\mathsf{x},\mathsf{w}\}$ and S be an X-nabla structure. For every $s,u \in \Sigma^*$, $\alpha : s \overset{X}{\hookrightarrow} u$ and $\nu \in \mathsf{Env}_s$, we have:*

$$S,\nu \models^X_s F \iff S,\mathrm{old}_\alpha(\nu) \models^X_u F$$

In other words, if we define $S[\![F]\!]$ as the subset of $S[\![\tau_1,\dots,\tau_k]\!]$ on which F is satisfied, our lemma states that $S[\![F]\!]$ is an X-nabla relation. The satisfaction relation also satisfies the expected properties with respect to substitution and unused variables.

Proposition 4.4. *For every X-nabla structure S, level s and assignment ν, we have:*

- *For every formula F, we have $S,\nu \models_s F[x := t]$ iff $S,\nu[x \mapsto S[\![t]\!]_s\nu] \models_s F$.*
- *For every F, $x_\tau \notin \mathrm{fv}(F)$ and $d,d' \in S[\![\tau]\!]_s$ we have $S,\nu[x \mapsto d] \models_s F$ iff $S,\nu[x \mapsto d'] \models_s F$.*

We say that a formula F is *X-valid* when for every X-nabla structure S, every level s and assignment ν, we have $S,\nu \models^X_s F$. For instance, the formula $\nabla x.F \Leftrightarrow F$ is X-valid when $\mathsf{w} \in X$ and $x \notin \mathrm{fv}(F)$. When $\mathsf{w} \in X$, the formula $\forall x.F \Rightarrow \nabla x.F$ is also X-valid.

We say that two formulas F and G are *X-equivalent* when the formula $F \Leftrightarrow G$ is X-valid, and we write $F \equiv_X G$ in this case. We omit X when it is irrelevant.

Proposition 4.5. *Let F and G be arbitrary formulas.*

1. *$\nabla x:\tau.(F \Rightarrow G) \equiv_X (\nabla x:\tau.F) \Rightarrow (\nabla x:\tau.G)$ for any X.*
2. *$\nabla x:\tau.\forall y:\theta.F \equiv_X \forall h:\tau\to\theta.\nabla x:\tau.F[y := hx]$ for any $X \neq \{\mathsf{w}\}$.*

The second equivalence crucially relies on the fact that nabla structures have enough maps. The difficulty is that, when considering $\nabla x:\tau.\forall y:\theta.F$ from level s, the universal quantification considers all possible values for y at any level u such that $s\tau \hookrightarrow u$. In particular, if $\mathsf{w} \in X$, the new level u may be of the form $s\tau v$. However, in $\forall h:\tau\to\theta.\nabla x:\tau.F[y := hx]$ the universal quantification (combined with the enough maps property) only covers values in levels of the form $sw\tau$. When $\mathsf{x} \in X$, we can take $w = v$ and swap it to the right position to obtain $s\tau v$. When $X = \{\mathsf{w}\}$, there is actually no hope to fix the issue: to see that the equivalence fails, take $F = P(x,y)$ and consider a structure S where P expresses that y only contains elements allocated after x; this is allowed by the condition saying that predicate interpretations should be w-relations, and falsifies the equivalence.

Proof sketch of proposition 4.5. We sketch the interesting case, which is the right-to-left direction of item 2. Assume. $S, \nu \models_s \forall h : \tau \to \theta. \nabla x : \tau. F[y := h\ x]$. Let us establish that $S, \nu \models_s \nabla x : \tau. \forall y : \theta. F$. We thus consider an arbitrary $\alpha : s\tau \hookrightarrow u$ and $d \in S[\![\theta]\!]_u$, and need to show the following:

$$S, \mathrm{old}_\alpha(\mathrm{old}_{s \hookrightarrow s\tau}(\nu)[x \mapsto \mathrm{new}_{s\tau}])[y \mapsto d] \models_u F$$

When $X = \emptyset$, the maps α and old_α are identities, we have $u = s\tau$, and we conclude easily because nabla structure have enough maps: d can be written as a function from level s applied to $\mathrm{new}_{s\tau}$. When $X = \{\mathsf{x}\}$, the map α has an inverse α^{-1} and our goal is equivalent to

$$S, \mathrm{old}_\alpha(\mathrm{old}_{s \hookrightarrow s\tau}(\nu)[x \mapsto \mathrm{new}_{s\tau}, y \mapsto \mathrm{old}_{\alpha^{-1}}(d)]) \models_u F$$

and then, by lemma 4.3, to the following, which can be proved as before:

$$S, \mathrm{old}_{s \hookrightarrow s\tau}(\nu)[x \mapsto \mathrm{new}_{s\tau}, y \mapsto \mathrm{old}_{\alpha^{-1}}(d)] \models_{s\tau} F$$

When $X = \{\mathsf{w}, \mathsf{x}\}$ we can write α as $\gamma \circ \beta$ where $\beta : s\tau \hookrightarrow s\tau v$ is a canonical injection and $\gamma : s\tau v \hookrightarrow u$ is bijective. In other words, β introduces new elements in the level after which γ re-orders elements. We can invert γ as we did for α above, and our goal becomes:

$$S, \mathrm{old}_{s\tau \hookrightarrow s\tau v}(\mathrm{old}_{s \hookrightarrow s\tau}(\nu)[x \mapsto \mathrm{new}_{s\tau}])[y \mapsto \mathrm{old}_{\gamma^{-1}}(d)] \models_{s\tau v} F$$

Using again lemma 4.3, this is equivalent to the following:

$$S, \mathrm{old}_{s\tau \hookrightarrow sv\tau}(\mathrm{old}_{s \hookrightarrow s\tau}(\nu)[x \mapsto \mathrm{new}_{s\tau}])[y \mapsto \mathrm{old}_{s\tau v \hookrightarrow sv\tau}(\mathrm{old}_{\gamma^{-1}}(d))] \models_{sv} F$$

We can finally conclude using our assumption at level sv, with the above assignment (before the addition of y) and using the *enough maps* property of S on value given to y above. □

Once this is established, one may verify that the sequent calculi $\mathrm{FO}\lambda^\nabla_X$ are sound with respect to our semantics for all $X \neq \{\mathsf{w}\}$.

Theorem 4.6 (Soundness)**.** *Let $X \subseteq \{\mathsf{w}, \mathsf{x}\}$ with $X \neq \{\mathsf{w}\}$. If the sequent $\Gamma \vdash \Delta$ is derivable in $FO\lambda^\nabla_X$, then $|\Gamma \vdash \Delta|$ is X-valid.*

The calculus $\mathrm{FO}\lambda^\nabla_{\mathsf{w}}$ is in fact unsound with respect to w-validity, because it proves the second equivalence of proposition 4.5, which we have shown to be invalid for $X = \{\mathsf{w}\}$. To resolve this mismatch, one may attempt to adapt either the semantics or the proof system. We would opt for the latter because, as discussed in section 2, it seems quite possible that $\mathrm{FO}\lambda^\nabla_{\mathsf{w}}$ does not stand proof-theoretically, while our semantics seems very natural.

5 Completeness

We show in this section that completeness also holds for our three logics. Because our calculi are cut-free, this implies as usual that the cut rule is admissible: adding it does not allow to derive more sequents.

Theorem 5.1. *For all $X \neq \{\mathsf{w}\}$, all X-valid sequents are derivable in $FO\lambda^\nabla_X$.*

For the rest of this section, we fix $X \neq \{\mathsf{w}\}$. We adapt the proof of [5], which follows a standard Henkin completeness argument: given an unprovable sequent, we build a counter-model for the corresponding formula. We work with marked judgements of the form $+J$ or $-J$, where J is a generic judgment as in our sequent calculi. In particular, we see a sequent $J_1, \ldots, J_n \vdash J'_1, \ldots, J'_m$ as the set of marked judgements $\{+J_1, \ldots, +J_n, -J'_1, \ldots, -J'_m\}$. When J and J' are judgements, we write $J \approx_X J'$ when this can be derived in our calculus. This is naturally extended to marked judgements.

Definition 5.2. A theory $\mathcal{T}$ is a set of marked judgements. It is said to be X-*consistent* when there is no sequent $J_1, \ldots, J_n \vdash J'_1, \ldots, J'_m$ with $\{+J_1, \ldots, +J_n, -J'_1, \ldots, -J'_m\} \subseteq \mathcal{T}$ that is provable in $\mathrm{FO}\lambda^{\nabla}_X$. It is X-inconsistent otherwise.

As a result, a consistent theory cannot contain two opposite judgements $+J$ and $-J$. It also cannot contain $+\sigma \rhd \bot$. Next, we adapt the notion of Hintikka theory from [5]; the formulation is almost identical but item 1 incorporates X adequately.

Definition 5.3. An X-consistent theory $\mathcal{T}$ is an X-*Hintikka theory* when:

1. If $J \approx_X J'$ and $J \in \mathcal{T}$ then $J' \in \mathcal{T}$;
2. If $+\sigma \rhd F \Rightarrow G \in \mathcal{T}$, then $-\sigma \rhd F \in \mathcal{T}$ or $+\sigma \rhd G \in \mathcal{T}$;
3. If $-\sigma \rhd F \Rightarrow G \in \mathcal{T}$, then $+\sigma \rhd F \in \mathcal{T}$ and $-\sigma \rhd G \in \mathcal{T}$;
4. If $+\sigma \rhd \forall x : \tau.F \in \mathcal{T}$, then $+\sigma \rhd F[x := t] \in \mathcal{T}$ for every term $t : \tau$;
5. If $-\sigma \rhd \forall x : \tau.F \in \mathcal{T}$, then there is a variable $h : \sigma \to \tau \notin \sigma$ such that $-\sigma \rhd F[x := h\sigma] \in \mathcal{T}$;
6. If $+\sigma \rhd \nabla x : \tau.F \in \mathcal{T}$, then $+(\sigma, x : \tau) \rhd F \in \mathcal{T}$;
7. If $-\sigma \rhd \nabla x : \tau.F \in \mathcal{T}$, then $-(\sigma, x : \tau) \rhd F \in \mathcal{T}$.

The two following lemmas are also adapted from [5]. Their proof is similar, which is unsurprising given that our sequent calculi only differ in the $(\approx)$ rule.

Lemma 5.4. *Let $\mathcal{T}$ be an X-consistent theory, F, G be formulas and J, J' be marked judgements. The following holds:*

1. *If $+\sigma \rhd F \Rightarrow G \in \mathcal{T}$, then $\mathcal{T} \cup \{-\sigma \rhd F\}$ or $\mathcal{T} \cup \{+\sigma \rhd G\}$ is X-consistent;*
2. *If $-\sigma \rhd F \Rightarrow G \in \mathcal{T}$, then $\mathcal{T} \cup \{+\sigma \rhd F\}$ and $\mathcal{T} \cup \{-\sigma \rhd G\}$ are X-consistent;*
3. *If $+\sigma \rhd \forall(x : \tau).F \in \mathcal{T}$, then for every λ-term t of type τ, $\mathcal{T} \cup \{+\sigma \rhd F[x := t]\}$ is X-consistent;*
4. *If $-\sigma \rhd \forall(x : \tau).F \in \mathcal{T}$, then there is a variable $h : \sigma \to \tau$ fresh with respect to $\mathcal{T}$ and σ such that $\mathcal{T} \cup \{-\sigma \rhd F[x := h\sigma]\}$ is X-consistent;*
5. *If $+\sigma \rhd \nabla(x : \tau).F \in \mathcal{T}$, then $\mathcal{T} \cup \{+(\sigma, x : \tau) \rhd F\}$ is X-consistent;*
6. *If $-\sigma \rhd \nabla(x : \tau).F \in \mathcal{T}$, then $\mathcal{T} \cup \{-(\sigma, x : \tau) \rhd F\}$ is X-consistent;*
7. *If $J \in \mathcal{T}$ and $J \approx_X J'$, then $\mathcal{T} \cup \{J'\}$ is X-consistent.*

The next lemma allows one to complete a consistent theory into a Hintikka theory, using the one just stated. The finiteness hypothesis is necessary because of item 4 above: if $\mathcal{T}$ is infinite, one may not find a fresh variable h to instantiate the lemma.

Lemma 5.5. *Every finite X-consistent theory is contained in an X-Hintikka theory.*

The last step consists in proving that every X-Hintikka theory has a model. The model we exhibit is an X-nabla structure on the enriched nabla universe of terms, naturally adapted from the term universe of [5] and similar to example 3.2. The following lemma justifies this construction. If $\sigma = (x_1 : \tau_1, \dots, x_n : \tau_n)$, we let ρ_σ be the substitution mapping x_i to $a^i_{\tau_i}$, the nominal constant representing the i^{th} generic variable when it has type τ_i.

Lemma 5.6. *Let $\mathcal{T}$ be an X-Hintikka theory. Let T be the nabla universe over terms, equipped with the least predicate interpretations $T[\![P]\!]$ such that, for any $+\sigma \rhd P(t_1, \dots, t_n) \in \mathcal{T}$, if we let $s = \tau_1, \dots, \tau_n$, we have $(t_1\rho_\sigma, \dots, t_n\rho_\sigma) \in T[\![P]\!]s$. The nabla universe T defined in this way is an X-nabla structure.*

The content of the previous statement is that the interpretations $T[\![P]\!]$ thus defined are actually X-nabla relations, which essentially comes from the $\approx_X$ condition on X-Hintikka theories.

Lemma 5.7. *Let $\mathcal{T}$ be an X-Hintikka theory. Let T be the X-nabla structure of lemma 5.6. For every $+J \in \mathcal{T}$, we have $T, \mathrm{id} \models_\varepsilon J$. For every $-J \in \mathcal{T}$, we have $T, \mathrm{id} \not\models_\varepsilon J$.*

Proof sketch. We prove both statements by a simultaneous induction on the formula within J. We sketch some key cases only.

- Assume $J = \sigma \rhd P(t_1, \dots, t_n)$.
  If $+\sigma \rhd P(t_1, \dots, t_n) \in \mathcal{T}$ then by definition, $(t_1\rho_\sigma, \dots, t_n\rho_\sigma) \in T[\![P]\!]_s$. Then $T, \rho_\sigma \models_s P(t_1, \dots, t_n)$, i.e. $T, \mathrm{id} \models_\varepsilon \nabla\sigma.P(t_1, \dots, t_n)$ by definition.
  If $-\sigma \rhd P(t_1, \dots, t_n) \in \mathcal{T}$, we cannot have $(t_1\rho_\sigma, \dots, t_n\rho_\sigma) \in T[\![P]\!]_s$ otherwise $\mathcal{T}$ would be inconsistent. Hence the result.
- Assume $+\sigma \rhd \forall x : \tau.F \in \mathcal{T}$, with $\sigma = (x_1 : \tau_1, \dots, x_n : \tau_n)$. We only show the argument for $X = \mathsf{wx}$; the others can be adapted. We want to show that $T, \mathrm{id} \models_\varepsilon \nabla\sigma.\forall x : \tau.F$, which is equivalent to $T, \mathrm{id} \models_\varepsilon \forall h : \sigma \to \tau.\nabla\sigma.F[x := h\sigma]$. We thus consider some arbitrary $\alpha : \varepsilon \hookrightarrow s$ and $t \in T[\![\sigma \to \tau]\!]_s$. By definition, this means that if a^i_τ appears in t then $\tau = s_i$. Hence if we take $\sigma' = (x'_1 : s_1, \dots, x'_n : s_n)$ with fresh variables x_i, there is a λ-term t' such that $t = t'\rho_{\sigma'}$ — concretely, t' is obtained by replacing $a^i_{s_i}$ by x'_i in t. We thus have:

$$\begin{aligned} T, \mathrm{old}_\alpha(\mathrm{id})[h \mapsto t] \models_s \nabla\sigma.F[x := h\sigma] &\iff T, \mathrm{id}[h \mapsto t'\rho_{\sigma'}] \models_s \nabla\sigma.F[x := h\sigma] \\ (\sigma' \text{ is free in } F) &\iff T, \rho_{\sigma'}[h \mapsto t'\rho_{\sigma'}] \models_s \nabla\sigma.F[x := h\sigma] \\ &\iff T, \rho_{\sigma'} \models_s \nabla\sigma.F[x := t'\sigma] \\ &\iff T, \mathrm{id} \models_\varepsilon \nabla\sigma'\sigma.F[x := t'\sigma] \end{aligned}$$

  Because $\mathcal{T}$ is a wx-Hintikka theory, $+(\sigma'\sigma) \rhd \forall x.F \in \mathcal{T}$ and thus $+\sigma'\sigma \rhd F[x := t'\sigma] \in \mathcal{T}$. We conclude by induction hypothesis.
  Now assume $-\sigma \rhd \forall x.F \in \mathcal{T}$. To show that $T, \mathrm{id} \not\models_\varepsilon \forall x.F$, it suffices to find a word s, a morphism $\alpha : \varepsilon \hookrightarrow s$ and a nominal term $t \in T[\![\tau]\!]_s$ such that $T, \mathrm{id}[h \mapsto t] \not\models_s F[x := h\sigma]$. But since $\mathcal{T}$ is a wx-Hintikka theory, we know that $-\sigma \rhd F[x := h\sigma] \in \mathcal{T}$ for a certain variable h not appearing in σ. Therefore we can instantiate s by ε and t by h to conclude by induction hypothesis. □

We can finally conclude our completeness argument.

Proof of theorem 5.1. Let $\Gamma \vdash \Delta$ be an X-unprovable sequent. Let $\mathcal{T}_0 = \{+J \mid J \in \Gamma\} \cup \{-J' \mid J' \in \Delta\}$. It is an X-consistent theory, otherwise $\Gamma \vdash \Delta$ would be provable. Thus, there is an X-Hintikka theory $\mathcal{T}$ such that $\mathcal{T}_0 \subseteq \mathcal{T}$. By the previous lemma, the X-nabla structure T is such that $T, \mathrm{id} \not\models_\varepsilon |\Gamma \vdash \Delta|$, which concludes the proof. □

6 Conclusion

We have generalized Goubault-Larrecq's semantics for Miller and Tiu's generic quantification [6] in two ways: first, we have lifted the restriction to a single base type by introducing *enriched* nabla sets; second, we have given a sound and complete semantics using these tools for Tiu's LG [10]. The crucial ingredients to make this work are to interpret formulas as wx-relations, which is made possible by taking a semantics of $\forall$ that considers possible extensions of the current level.

More broadly, we have raised the question of seriously considering substructural generic quantification. We have proposed a first definition of what this could mean through our modular sequent calculi $\mathrm{FO}\lambda^\nabla_X$. However, these calculi correspond to our semantics only for $X \neq \{\mathsf{w}\}$. To circumvent this issue, we are currently investigating infinitarily branching proof systems, which appear to provide an exact match (i.e., soundness and completeness) for all X. Furthermore, for all $X \neq \{\mathsf{w}\}$, the infinitary calculus for X shows the same sequents as $\mathrm{FO}\lambda^\nabla_X$.

On the proof-theoretical side, the obvious direction for future work would be to closely study the relationship between $\mathrm{FO}\lambda^\nabla_{\mathsf{wx}}$ and LG, and whether LG's cut elimination can be adapted to all $\mathrm{FO}\lambda^\nabla_X$ logics. One might also wonder whether the semantics can be adapted to match the intuitionistic versions of those calculi; nabla sets can likely be combined with Kripke semantics in a mostly orthogonal fashion.

Finally, one might wonder whether nominal sets could provide an appropriate semantical framework for (substructural) generic quantification, as an alternative to nabla sets. One reason to think that it should be the case is that Goubault-Larrecq's nabla sets are almost identical to the Schanuel topos, which is known to be equivalent to nominal sets [8]. There is a slight difference, in that the Schanuel topos requires, intuitively, that elements $a \in D_u$ of support s can always be written as $\mathrm{old}_{s \hookrightarrow u}(b)$ for $b \in D_s$. This condition can be formally stated in the context of nabla sets, and also for our enriched nabla sets. Moreover, nominal sets can also be naturally enriched with names of several types. One might then expect that the known equivalence extends, so that enriched nabla sets with the extra condition are equivalent to enriched nominal sets; but none of our attempts to do so ended up being successful. Hence, our answer so far would be that enriched nabla sets provide a more flexible framework than enriched nominal sets, though this issue certainly deserves to be treated in more details.

References

[1] David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu & Yuting Wang (2014): *Abella: A System for Reasoning about Relational Specifications. J. Formaliz. Reason.* 7(2), pp. 1–89.

[2] David Baelde, Adrien Koutsos & Joseph Lallemand (2023): *A higher-order indistinguishability logic for cryptographic reasoning.* In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, pp. 1–13.

[3] Miko1.8aj Bojańczyk, Bartek Klin & S1.8awomir Lasota (2014): *Automata theory in nominal sets. Logical Methods in Computer Science* 10.

[4] Murdoch J. Gabbay & Andrew M. Pitts (2002): *A New Approach to Abstract Syntax with Variable Binding. Formal Aspects of Computing* 13(3–5), p. 341–363, doi:10.1007/s001650200016. Available at `http://dx.doi.org/10.1007/s001650200016`.

[5] Jean Goubault-Larrecq (2019): *A semantics for nabla. Mathematical Structures in Computer Science* 29(8), p. 1250–1274, doi:10.1017/S0960129518000063.

[6] Dale A. Miller & Alwen F. Tiu (2003): *A Proof Theory for Generic Judgements: An extended abstract*, pp. 118–127.

[7] Andrew M Pitts (2003): *Nominal logic, a first order theory of names and binding. Information and computation* 186(2), pp. 165–193.

[8] Andrew M Pitts (2013): *Nominal sets: Names and symmetry in computer science*. Cambridge University Press.

[9] Ulrich Schöpp (2007): *Modelling Generic Judgements. Electronic Notes in Theoretical Computer Science* 174(5), p. 19–35, doi:10.1016/j.entcs.2007.01.027. Available at `http://dx.doi.org/10.1016/j.entcs.2007.01.027`.

[10] Alwen Tiu (2006): *A Logic for Reasoning about Generic Judgments*. In: *LFMTP@FLoC*, *Electronic Notes in Theoretical Computer Science* 5, Elsevier, pp. 3–18.

[11] Christian Urban (2008): *Nominal techniques in Isabelle/HOL. Journal of Automated Reasoning* 40(4), pp. 327–356.