Nominal Type Theory by Nullary Internal Parametricity *

Antoine Van Muylder, Andreas Nuyts, and Dominique Devriese

DistriNet, KU Leuven, Belgium

Nominal syntax Nominal data types are data types in which binders of the object language are typed using a special type former called name abstraction, which we write $@\mathsf{N} \multimap -$ and which has been axiomatized in various ways. Intuitively $a' : @\mathsf{N} \multimap A$ is a term of A that may mention freely a freshly chosen name. Nominal data types can also feature a type of names Nm in their signature. The following data type is the nominal syntax of the untyped λ-calculus.

```
data Ltm (j : nat) : 𝒰
  env : Fin j → Ltm j
  var : Nm → Ltm j
  app : Ltm j → Ltm j → Ltm j
  lam : (@N ⊸ Ltm j) → Ltm j
```

```
indLtm : (P : Ltm j → 𝒰) → (∀k. P(env k)) →
  ((n : Nm) → P(var n)) →
  (∀a b. P a → P b → P(app a b)) →
  (∀g. ((x:@N) ⊸ P(g x)) → P(lam g)) →
  ∀t.P t
```

Nominal data types can not be expressed in standard type theory and require a stronger metalanguage. Yet they offer several advantages as a representation of syntax. First, α-equivalent terms are definitionally equal, a consequence of how $@\mathsf{N} \multimap -$ is axiomatized. Second, nominal data types have a (closed) induction principle (e.g., `indLtm`). Third, by contrast with the De Bruijn representation, weakening by a name simply means extending the context with that name, and all programs in the metalanguage are automatically stable under this operation. Fourth, formal reasoning with nominal syntax can sometimes match on-paper reasoning [Urb08].

Nominal frameworks Nominal frameworks [SS04, Che12, SPG03, PMD14, Pit03] are metalanguages attempting to support fresh name binding and ideally nominal data types. Nominal frameworks can decide to axiomatize the name abstraction type former in either a positive (notation $@\mathsf{N} \cdot -$) or in a negative fashion (notation $@\mathsf{N} \multimap -$). The two presentations are semantically equivalent [SS04] but syntactically have different pros and cons.

If the positive presentation is used, a name abstraction $a' : @\mathsf{N} \cdot A$ can be understood as a *pair up to α-renaming* $a' = \langle x, a\rangle$ where x is bound in a, i.e. x is fresh in $\langle x, a\rangle$. This presentation is convenient since the user is allowed to deeply *pattern match* on such binding name-term pairs, an ability that we call nominal pattern matching. For example defining a function f out of the $\mathsf{Ltm}\, j$ data type would involve writing a pattern matching clause like $f(\mathtt{lam}\,\langle x, t\rangle) = \mathtt{rhs}(x, t)$ where x can be mentioned in `rhs`. However the typing rules of the positive presentation are inconvenient because one can only eliminate a pair $\langle x, t\rangle$ into a term/type for which x is fresh. Intuitively this is to prevent $\mathtt{rhs}(x, t) = x$ which outputs different results for α-equivalent pairs.

If the negative presentation is used, a name abstraction $a' : @\mathsf{N} \multimap A$ is rather treated as an affine *function*, i.e. a function from the $@\mathsf{N}$ (pre)type with an extra side condition asserting that it can only consume fresh variables ($x : @\mathsf{N}$). The inference rules (see fig. 1) of this presentation are straightforward to specify, yet the ability to pattern match is seemingly lost.

Contributions We propose[1] nullary (0-ary) internally parametric type theory (nullary PTT) as the foundation for nominal dependent type theory (nominal TT). One way to understand n-ary PTT is to compare it to cubical type theory. While cubical type theory is a language intepreting in higher groupoids, n-ary PTT is a language intepreting in higher n-ary reflexive graphs.

Our nominal TT is obtained as a modest extension of Cavallo and Harper's (univalent and) parametric type theory [CH21] (CH type theory) where we replace the arity $n = 2$ by $n = 0$. In

*This is an abstract for https://arxiv.org/pdf/2512.09464, submitted to the TYPES25 post-proceedings.

[1]The idea existed in the community [ND24, Nar25].

short, nominal TT = nullary PTT + name induction + nominal data types. More precisely, firstly, we show that nullary PTT in itself already supports important features of nominal frameworks: (1) Negative name abstraction types have exactly the same rules as nullary bridge types. (2) A freshness type former is given by the Gel type former, which in general turns n-ary relations into n-ary bridges in the universe. Indeed, for $n = 0$, if $(x : @\mathsf{N}) \in \Gamma$, then $\Gamma \vdash \mathsf{Gel}\, A\, x$ can be understood as "the A's for which x is fresh" (see GelI rule from fig. 1). (3) Additionally we can implement positive name abstractions, a name swapping operation and the local-scoping primitive ν of [PMD14] (roughly, the latter primitive is used to witness freshness). Secondly, to express nominal data types we need a proper, cartesian type of names $\vdash \mathsf{Nm}$. It comes equipped with a novel dependent eliminator $\mathsf{ind}_{\mathsf{Nm}}$ called *name induction*, which expresses for a term $\Gamma \vdash n : \mathsf{Nm}$ and a bridge variable x in Γ, that n is either just x, or x is *fresh* in n. In the rule for $\mathsf{ind}_{\mathsf{Nm}}$, freshness is expressed typally using a Gel type. Finally, we allow nominal data types such as $\mathsf{Ltm}\, j$ above, whose constructors can take recursive arguments in which new, fresh names are bound.

Nominal TT is sound as it interprets in the expected bicubical set model (presheaves over (nullary) affine cubes × cartesian cubes, similar to CH). There is currently no implementation of the theory tough such an implementation could be quite similar to [VND24].

$$\frac{\Gamma, (x : @\mathsf{N}) \vdash A\ \mathsf{type}}{\Gamma \vdash (x : @\mathsf{N}) \multimap A\ \mathsf{type}}\ \multimap\mathrm{F} \qquad \frac{\Gamma, (x : @\mathsf{N}) \vdash a : A}{\Gamma \vdash \lambda\, x.\, a : (x : @\mathsf{N}) \multimap A}\ \multimap\mathrm{I}$$

$$\frac{(x : @\mathsf{N}) \in \Gamma \qquad \Gamma \setminus x \vdash a' : (y : @\mathsf{N}) \multimap A}{\Gamma \vdash a'\, x : A[x/y]}\ \multimap\mathrm{E} \qquad \frac{(x : @\mathsf{N}) \in \Gamma \qquad \Gamma \setminus x, (y : @\mathsf{N}) \vdash a : A}{\Gamma \vdash (\lambda\, y.\, a)\, x = a[x/y] : A[x/y]}\ \multimap\beta$$

$$\frac{\Gamma, (x : @\mathsf{N}) \vdash a'_0\, x = a'_1\, x : A}{\Gamma \vdash a'_0 = a'_1 : (x : @\mathsf{N}) \multimap A}\ \multimap\eta \qquad \frac{(x : @\mathsf{N}) \in \Gamma \qquad \Gamma \setminus x \vdash A\ \mathsf{type}}{\Gamma \vdash \mathsf{Gel}\, A\, x\ \mathsf{type}}\ \textsc{GelF}$$

$$\frac{(x : @\mathsf{N}) \in \Gamma \qquad \Gamma \setminus x \vdash a : A}{\Gamma \vdash \mathsf{gel}\, a\, x : \mathsf{Gel}\, a\, x}\ \textsc{GelI} \qquad \frac{\Gamma, (x : @\mathsf{N}) \vdash g : \mathsf{Gel}\, A\, x}{\Gamma \vdash \mathsf{ung}\, (\lambda\, x.\, g) : A}\ \textsc{GelE} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{ung}\, (\lambda\, x.\, \mathsf{gel}\, a\, x) = a}\ \textsc{Gel}\beta$$

$$\frac{\textsc{GelF prem.} \qquad \Gamma \setminus x, (y : @\mathsf{N}) \vdash g_0, g_1 : \mathsf{Gel}\, A\, y \qquad \Gamma \vdash \mathsf{ung}(\lambda y.\, g_0)\ =\ \mathsf{ung}(\lambda y.\, g_1) : A}{\Gamma \vdash g_0[x/y] = g_1[x/y] : \mathsf{Gel}\, A\, x}\ \textsc{Gel}\eta \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{Nm}\ \mathsf{type}}\ \mathsf{Nm}\mathrm{F} \qquad \frac{(x : @\mathsf{N}) \in \Gamma}{\Gamma \vdash \mathsf{c}\, x : \mathsf{Nm}}\ \mathsf{Nm}\mathrm{I}$$

$$\frac{(x : @\mathsf{N}) \in \Gamma \qquad \Gamma \vdash n : \mathsf{Nm} \qquad \Gamma, z : \mathsf{Nm} \vdash B\ \mathsf{type} \qquad \Gamma \vdash b_0 : B[\mathsf{c}\, x/z] \qquad \Gamma, (g : \mathsf{Gel}\, \mathsf{Nm}\, x) \vdash b_1 : B[\mathsf{forget}\, x\, g/z] \qquad \Gamma, (g : \mathsf{Gel}\, \mathsf{Nm}\, x) \vdash \mathsf{forget}\, x\, g := \mathsf{ext}\, (\lambda\, g'\, y.\, \mathsf{ung}\, g')\, x\, g : \mathsf{Nm}}{\Gamma \vdash \mathsf{ind}_{\mathsf{Nm}}\, x\, n\, b_0\, (\lambda\, g.\, b_1) : B[n/z]}\ \mathsf{Nm}\mathrm{E}$$

$$\frac{\mathsf{Nm}\mathrm{E}\ \text{prem. without}\ n}{\Gamma \vdash \mathsf{ind}_{\mathsf{Nm}}\, x\, (\mathsf{c}\, x)\, b_0\, (\lambda\, g.\, b_1) = b_0 : B[\mathsf{c}\, x/y]}\ \mathsf{Nm}\beta_0 \qquad \frac{\mathsf{Nm}\mathrm{E}\ \text{prem. without}\ n \qquad \Gamma \setminus x \vdash n : \mathsf{Nm}}{\Gamma \vdash \mathsf{ind}_{\mathsf{Nm}}\, x\, n\, b_0\, (\lambda\, g.\, b_1) = b_1[\mathsf{gel}\, n\, x/g] : B[n/z]}\ \mathsf{Nm}\beta_1$$

Figure 1: Bridge, Gel and Nm. $\Gamma \setminus x$ removes from Γ the variable x and everything to its right. The forget function in NmE uses the extent primitive ext from nullary PTT (not presented).

Using the theory A useful set of theorems to reason about nominal syntax in nominal TT is our Structure Abstraction Principle (SAP). The SAP at a (dependent) type A explains how the Bridge type former and A commute. Here are some proved instances of the SAP: $(@\mathsf{N} \multimap (A \to B)) \simeq (@\mathsf{N} \multimap A) \to (@\mathsf{N} \multimap B)$, $(@\mathsf{N} \multimap \mathcal{U}) \simeq \mathcal{U}$, $(@\mathsf{N} \multimap \mathsf{Nm}) \simeq 1 + \mathsf{Nm}$, $(@\mathsf{N} \multimap \mathsf{Ltm}\, j) \simeq \mathsf{Ltm}\, (j + 1)$.

Programming and reasoning with nominal syntax in full generality is possible in nominal TT, thanks to the presence of nominal recursion/induction (see e.g. indLtm). In particular, from the type of nominal recursion, defining a function f by matching (non-deeply) on a binder involves recursively calling f *under its "intervalic" parametricity* $(@\mathsf{N} \multimap f) := \lambda d'\, x.\, f(d'x) : (@\mathsf{N} \multimap D) \to @\mathsf{N} \multimap E$ (and often using the SAP at E to proceed). Since the definition of f uses $@\mathsf{N} \multimap f$, proving f correct often involves proving that $@\mathsf{N} \multimap f$ is propositionally equal to another term obtained recursively from f, the observational parametricity (i.e. param. translation) of f.

Nominal induction can emulate nominal pattern matching (up to prop. equality). Matching on a pattern with binder depth n involves using the n-fold iterated parametricity of f. In observational nominal TT, nominal pattern matching would be recovered up to definitional equality.

References

[CH21] Evan Cavallo and Robert Harper. Internal parametricity for cubical type theory. *Log. Methods Comput. Sci.*, 17(4), 2021. URL: https://doi.org/10.46298/lmcs-17(4:5)2021, doi:10.46298/LMCS-17(4:5)2021.

[Che12] James Cheney. A dependent nominal type theory. *Log. Methods Comput. Sci.*, 8(1), 2012. doi:10.2168/LMCS-8(1:8)2012.

[Nar25] Narya development team. Nominal syntax — narya documentation, nov 2025. Accessed: 2025-11-19. URL: https://narya.readthedocs.io/en/latest/nominal.html.

[ND24] Andreas Nuyts and Dominique Devriese. Transpension: The right adjoint to the Pi-type. *Log. Methods Comput. Sci.*, 20(2), 2024. URL: https://doi.org/10.46298/lmcs-20(2:16)2024, doi:10.46298/LMCS-20(2:16)2024.

[Pit03] Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Inf. Comput.*, 186(2):165–193, 2003. doi:10.1016/S0890-5401(03)00138-X.

[PMD14] Andrew M. Pitts, Justus Matthiesen, and Jasper Derikx. A dependent type theory with abstractable names. In Mauricio Ayala-Rincón and Ian Mackie, editors, *Ninth Workshop on Logical and Semantic Frameworks, with Applications, LSFA 2014, Brasília, Brazil, September 8-9, 2014*, volume 312 of *Electronic Notes in Theoretical Computer Science*, pages 19–50. Elsevier, 2014. URL: https://doi.org/10.1016/j.entcs.2015.04.003, doi:10.1016/J.ENTCS.2015.04.003.

[SPG03] Mark R. Shinwell, Andrew M. Pitts, and Murdoch Gabbay. FreshML: programming with binders made simple. In Colin Runciman and Olin Shivers, editors, *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming, ICFP 2003, Uppsala, Sweden, August 25-29, 2003*, pages 263–274. ACM, 2003. doi:10.1145/944705.944729.

[SS04] Ulrich Schöpp and Ian Stark. A dependent type theory with names and binding. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic*, pages 235–249, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30124-0_20.

[Urb08] Christian Urban. Nominal techniques in isabelle/hol. *J. Autom. Reason.*, 40(4):327–356, 2008. URL: https://doi.org/10.1007/s10817-008-9097-2, doi:10.1007/S10817-008-9097-2.

[VND24] Antoine Van Muylder, Andreas Nuyts, and Dominique Devriese. Internal and observational parametricity for cubical agda. *Proc. ACM Program. Lang.*, 8(POPL):209–240, 2024. doi:10.1145/3632850.