Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 312 (2015) 19–50

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Dependent Type Theory with Abstractable Names

Andrew M. Pitts[a,1,2], Justus Matthiesen [a,3,4] and Jasper Derikx[b,5]

[a] *Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, UK*

[b] *Radboud University, 6500 GL Nijmegen, Netherlands*

Abstract

This paper describes a version of Martin-Löf's dependent type theory extended with names and constructs for freshness and name-abstraction derived from the theory of nominal sets. We aim for a type theory for computing and proving (via a Curry-Howard correspondence) with syntactic structures which captures familiar, but informal, 'nameful' practices when dealing with binders.

Keywords: binding, dependent types, names, nominal sets

1 Introduction

We aim to develop a constructive version of nominal logic [15] as a dependent type theory. From a programming point of view we would like to combine Agda/Coq style theorem-proving (particularly inductively defined indexed families of types and dependent pattern-matching) with FreshML [21] style meta-programming for syntax with binding operations. Achieving these aims requires a constructive treatment of the nominal sets notion of *freshness* [16, Chapter 3]. Here we give one such treatment as an extension of Martin-Löf type theory.

The functional programming language FreshML is impure: it ensures freshness of names via generativity and (hence) avoids checking that a locally scoped name

[1] Partially supported by the UK EPSRC program grant EP/K008528/1, *Rigorous Engineering for Mainstream Systems* (REMS).

[2] Email: andrew.pitts@cl.cam.ac.uk

[3] Supported by the UK EPSRC leadership fellowship (Peter Sewell) grant EP/H005633/1, *Semantic Foundations for Real-World Systems*.

[4] Email: justus.matthiesen@cl.cam.ac.uk

[5] Email: jasperderikx@gmail.com

http://dx.doi.org/10.1016/j.entcs.2015.04.003

does not occur in the support of the meaning of the expression in which it is used. The original version of the language, 'FreshML 2000' [17], attempted to carry out such checks by inferring freshness information as part of the type system, but was found to be too restrictive in the context of a Turing-powerful language – the main difficulty being how to decide whether a name n is fresh for a (higher-order) function f, written $n \# f$. Within nominal sets [16], the definition of the freshness relation involves quantification over finite sets of names: $n \# f$ means that there exists a finite set of names supporting f that does not contain the name n. In practice, one often relies upon the fact that this relation is invariant under permuting names and uses the following sound method that reflects on a concrete, meta-theoretic version of freshness, *viz.* non-occurrence:

> To prove $n \# f$, pick a name n' that does not occur in the current context (that is, one that is meta-theoretically fresh) and prove $(n\ \ n') \cdot f = f$, which in the presence of function extensionality, is equivalent to showing $(\forall x)\ (n\ \ n') \cdot (f\, x) = f((n\ \ n') \cdot x)$. (As usual, $(n\ \ n') \cdot x$ denotes the result of transposing names n and n' in an element x of a nominal set.) Since $n' \# f$ holds by choice of n', applying the permutation $(n\ \ n')$ that swaps n and n' we get $n = (n\ \ n') \cdot n' \# (n\ \ n') \cdot f = f$, as required.

This proof principle was adopted by nominal algebra [9]/nominal equational logic [5] and emphasised particularly in Clouston's thesis [4] and the recent work of Crole and Nebel [7], which both make freshness assertions

$$\Gamma \vdash n \,\#\, t : T$$

equivalent to equality judgements of the form

$$\Gamma[n' : N] \vdash (n\ \ n') \cdot t = t : T \tag{1}$$

where $(n\ \ n') \cdot \_$ is the (object-level) name-swapping operation, the context Γ contains hypotheses about freshness of names for free variables and $\Gamma[n : N]$ [6] adds to Γ an extra freshness hypotheses for a (meta-theoretically) new name n of some sort N. Equality jugements, such as (1), will be axiomatized by the type theory introduced in Sect. 2.

We call this delegation of freshness to definitional equality *definitional freshness*. It means that equality judgements get intertwined with typing judgements in an extra way from what already happens in dependently typed systems. The advantage of this approach is that we can give 'pure' versions of locally scoped names and concretion of name-abstractions with a semantics just using nominal sets, rather than, for example, nominal restriction sets [16, section 9.1]. The next section describes such a dependent type theory with abstractable names. Since it is an extension of Martin-Löf's Type Theory with many of the features of FreshML, we call it FreshMLTT. Section 3 describes the intended model of FreshMLTT; we

[6] Instead of using the 'flattened' contexts $\{\boldsymbol{n}_1 \# x_1 : T_1, \boldsymbol{n}_2 \# x_2 : T_2, \ldots\}$ from [24,9,5,4,7], here we will use 'bunched' ones, as in [19,18,3], because they fit better with the 'telescopic' nature of contexts in dependent type theory.

organise nominal sets into an instance of Dybjer's notion of category with families (CwF) [8] and develop a dependent version of the nominal sets notion of name abstraction. The interpretation of FreshMLTT in this CwF is given in Sect. 4, together with its soundness (Theorem 4.2). Section 5 surveys previous work on combining nominal sets with dependent types and Sect. 6 outlines what needs to be done to develop FreshMLTT further. In particular, the question as to whether FreshMLTT has decidable type-checking is open.

2 FreshMLTT

In this paper we consider an extension of Martin-Löf Type Theory [14] with names that can be swapped, compared for equality, locally scoped and abstracted. We call it FreshMLTT. The syntax of its expressions is given in Fig. 1. There are two sorts of bindable identifier, *variables* (x) and *names* (n), and expressions are identified up to α-equivalence: the binding forms are $\nu[n : N]\_$, $\mathsf{И}[n : N]\_$, $\Pi(x : T)\_$, $\alpha[n : N]\_$ and $\lambda(x : T)\_$. We write $\mathrm{fv}(e)$ for the finite set of free variables of an expression e and $\mathrm{fn}(e)$ for its finite set of free names. Capture-avoiding substitution of e for all free occurrences of a variable x in e' is denoted $e'(e/x)$; names are not subject to substitution (see Sect. 2.1). We write $\mathrm{dom}(\Gamma)$ for the finite set of variables and names that are declared in a context Γ. We adopt Agda-style notation for multiple context extensions and write $\Gamma[n\ n' : N]$ rather than $\Gamma[n : N][n' : N]$, for example.

The forms of judgement of FreshMLTT are given in Fig. 2. The rules for deriving valid instances of these judgements are listed in Appendix A. We discuss them in the following sections (2.1–2.3).

2.1 Names

FreshMLTT is intended to be used as a meta-language for describing, and computing with, the syntax and semantics of various languages. These 'object languages' typically involve various sorts of name (for example, names of variables, communication channels, etc.). Therefore FreshMLTT has a countably infinite collection $N, N', \ldots \in \mathit{Nsort}$ of distinguished types[7], called *name sorts*

$$\frac{\Gamma \vdash \mathrm{ok}}{\Gamma \vdash N}\ (\textsc{n-form})$$

the terms of which stand for object-language names. The usual rules of Martin-Löf Type Theory for extending a context

$$\frac{\Gamma \vdash T \qquad x \notin \mathrm{dom}(\Gamma)}{\Gamma(x : T) \vdash \mathrm{ok}}\ (\textsc{ctx-ext-v}) \qquad \frac{\Gamma \vdash \mathrm{ok} \qquad (x : T) \in \Gamma}{\Gamma \vdash x : T}\ (\textsc{v-intro})$$

[7] It would be natural to allow name sorts to be parameterised by the elements of other types, but for simplicity we do not do that here.

names $n \in \mathit{Name}$			(countably infinite set)
name sorts $N \in \mathit{Nsort}$			(countably infinite set)
variables $x \in \mathit{Var}$			(countably infinite set)
expressions $e \in \mathit{Exp}$	$::=$	$\Gamma \mid T \mid t$	
contexts $\Gamma \in \mathit{Ctx}$	$::=$	$\diamond$	empty
	$\mid$	$\Gamma(x : T)$	extend with variable
	$\mid$	$\Gamma[n : N]$	extend with fresh name
types $T \in \mathit{Type}$	$::=$	N	sort of names
	$\mid$	$(n\ \ n) \cdot T$	name swapping
	$\mid$	$\nu[n : N]\, T$	locally scoped name
	$\mid$	$\mathit{И}[n : N]T$	name-abstraction type
	$\mid$	$\Pi(x : T)T$	dependent function type
	$\ldots$		(other constructs of MLTT)
terms $t \in \mathit{Term}$	$::=$	x	variable
	$\mid$	n	name
	$\mid$	$(n\ \ n) \cdot t$	name swapping
	$\mid$	$\mathsf{if}\ t = n\ \mathsf{then}\ t\ \mathsf{else}\ t$	branch on name equality
	$\mid$	$\nu[n : N]\, t$	locally scoped name
	$\mid$	$\alpha[n : N]t$	name-abstraction
	$\mid$	$t \mathbin{@} n$	concretion
	$\mid$	$\lambda(x : T)t$	λ-abstraction
	$\mid$	$t\, t$	application
	$\ldots$		(other constructs of MLTT)

Fig. 1. Syntax of FreshMLTT

allow one to hypothesise a variable (that is, an unknown element) of some type T; and so when $T = N$, we hypothesise an unknown object-level name of sort N. However in FreshMLTT, in addition to variables $x, y, \ldots \in \mathit{Var}$, there is a disjoint kind of bindable identifier, meta-level *names* $n, m, \ldots \in \mathit{Name}$, together

Judgements take the form $\Gamma \vdash \mathcal{J}$, where

$$
\begin{array}{rll}
\mathcal{J} ::= & \mathrm{ok} & \text{well-formed context} \\
\mid & T & \text{well-formed type} \\
\mid & t : T & \text{typing relation} \\
\mid & (n : N) \,\#\, T & \text{definitional freshness for types} \\
\mid & (n : N) \,\#\, t : T & \text{definitional freshness for terms} \\
\mid & T = T & \text{definitionally equal types} \\
\mid & t = t : T & \text{definitionally equal terms}
\end{array}
$$

Fig. 2. Judgements of FreshMLTT

with associated context extension and introduction rules:

$$\frac{\Gamma \vdash \mathrm{ok} \qquad n \notin \mathrm{dom}(\Gamma)}{\Gamma[n : N] \vdash \mathrm{ok}}\ (\textsc{ctx-ext-n}) \qquad \frac{\Gamma \vdash \mathrm{ok} \qquad [n : N] \in \Gamma}{\Gamma \vdash n : N}\ (\textsc{n-intro})$$

The difference between a variable of type N and a name of type N is that the latter has a fixed, context-insensitive identity: if n and n' are different names, then the Boolean expression $n = n'$ is convertible to false; whereas if x and x' are different variables, then the Boolean expression $x = x'$ is neutral. For simplicity we do not introduce a type of Booleans explicitly and instead use test-and-branch expressions at each type

$$\frac{\Gamma \vdash t : N \qquad \Gamma \vdash n : N \qquad \Gamma \vdash t_1 : T \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash \mathsf{if}\ t = n\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : T}\ (\textsc{ifeq})$$

together with the definitional equalities

$$\left.\begin{array}{l} \mathsf{if}\ n = n\ \mathsf{then}\ t\ \mathsf{else}\ t' = t \\ \mathsf{if}\ n = n'\ \mathsf{then}\ t\ \mathsf{else}\ t' = t' \end{array}\right\}\ (\textsc{if-comp}) \qquad (\text{where } n \neq n').$$

Because of those equalities, the validity of the judgements of FreshMLTT is not preserved under the operation of substituting a term for a name, whereas substitution of terms for variables does preserve validity, as usual. Instead, names obey a weaker substitution discipline familiar from the work on nominal logic: validity of judgements is preserved under permutation of names and in particular, under the operation of swapping two names.

2.2 Definitional freshness

When computing with object languages involving names, the relation of not occurring free in a language expression is of crucial importance. In particular one frequently has to introduce names that are not free in the current context. This

is supported in FreshMLTT by the context extension rule (CTX-EXT-N) mentioned above: in the context $\Gamma[n : N]$ it is intended that any variables declared in Γ are restricted to range over object-level entities in which the name n does not occur free. The nominal sets relation of *freshness* ($a \# x$) [16, Chapter 3] provides a syntax-independent meaning for 'does not occur free', the constructive properties of which are formalised by FreshMLTT. To do so, there are expressions for the operation of swapping two names in types or terms

$$\frac{\Gamma \vdash n : N \qquad \Gamma \vdash n' : N \qquad \Gamma \vdash T}{\Gamma \vdash (n\ \ n') \cdot T}\ (\text{SWAP-TYP}) \qquad \frac{\Gamma \vdash n : N \qquad \Gamma \vdash n' : N \qquad \Gamma \vdash t : T}{\Gamma \vdash (n\ \ n') \cdot t : (n\ \ n') \cdot T}\ (\text{SWAP-TERM})$$

and definitional equalities

$$\frac{\Gamma \vdash T \qquad \Gamma\Delta \vdash \mathrm{ok} \qquad [n : N], [n' : N] \in \Delta}{\Gamma\Delta \vdash (n\ \ n') \cdot T = T}\ (\text{FRESH-HYP-TYP}) \qquad \frac{\Gamma \vdash t : T \qquad \Gamma\Delta \vdash \mathrm{ok} \qquad [n : N], [n' : N] \in \Delta}{\Gamma\Delta \vdash (n\ \ n') \cdot t = t : T}\ (\text{FRESH-HYP-TERM})$$

that formalize a basic property of freshness: $a \# x \wedge a' \# x \Rightarrow (a\ \ a') \cdot x = x$ [16, Proposition 3.1]. Another key property of freshness is that given any element x of a nominal set, there is some atom a with $a \# x$. A consequence of this are structural rules for eliminating unused fresh-name assumptions, taken from the work on nominal equational logic [5] and algebra [9]:

$$\frac{\Gamma\Delta \vdash T \qquad \Gamma\Delta \vdash T' \qquad \Gamma[n : N]\Delta \vdash T = T'}{\Gamma\Delta \vdash T = T'}\ (\text{N-ELIM-TYP}) \qquad \frac{\Gamma\Delta \vdash t : T \qquad \Gamma\Delta \vdash t' : T \qquad \Gamma[n : N]\Delta \vdash t = t' : T}{\Gamma\Delta \vdash t = t' : T}\ (\text{N-ELIM-TERM})$$

As explained in the Introduction, FreshMLTT can make judgements about whether a name is fresh for an expression via definitional equalities involving swapping with a name that is known to be fresh from the context:

$$\frac{\Gamma \vdash n : N \qquad \Gamma \vdash T \qquad \Gamma[n' : N] \vdash (n\ \ n') \cdot T = T}{\Gamma \vdash (n : N) \,\#\, T}\ (\text{FRESH-TYP}) \qquad \frac{\Gamma \vdash (n : N) \,\#\, T \qquad \Gamma[n' : N] \vdash (n\ \ n') \cdot t = t : T}{\Gamma \vdash (n : N) \,\#\, t : T}\ (\text{FRESH-TERM})$$

Such *definitional freshness* judgements are needed to express the properties of name abstraction, to which we turn in the next section.

Remark 2.1 Note that definitional freshness can hold even if a name n is not meta-theoretically fresh for an expression e, that is, even if n occurs free in e. For example n occurs free in the term $\mathsf{if}\ n = n'\ \mathsf{then}\ n\ \mathsf{else}\ n'$, but nevertheless

$$\diamond[n\ n' : N] \vdash (n : N) \,\#\, (\mathsf{if}\ n = n'\ \mathsf{then}\ n\ \mathsf{else}\ n') : N$$

is a valid judgement in FreshMLTT, because of the definitional equality (IF-COMP) mentioned above.

2.3 Abstracting and locally scoping names

The usual formation and introduction rules for dependent functions

$$\frac{\Gamma(x:T)\vdash T'}{\Gamma\vdash \Pi(x:T)T'}\ (\Pi\text{-FORM}) \qquad \frac{\Gamma(x:T)\vdash t':T'}{\Gamma\vdash \lambda(x:T)t':\Pi(x:T)T'}\ (\Pi\text{-INTRO})$$

discharge a variable from the context. In FreshMLTT there are formation and introduction rules for *name abstractions*, which involve discharging a fresh name from the context:

$$\frac{\Gamma[n:N]\vdash T}{\Gamma\vdash \text{И}[n:N]T}\ (\text{И}\text{-FORM}) \qquad \frac{\Gamma[n:N]\vdash t:T}{\Gamma\vdash \alpha[n:N]t:\text{И}[n:N]T}\ (\text{И}\text{-INTRO})$$

Name abstraction types can be used to indicate that a syntactical construct in some object language is a binder. For example, assuming FreshMLTT is augmented with data types for finite lists [14, Chapter 10], one might introduce a dependent type $\mathsf{Term}(\ell)$ for λ-terms whose free variables are in the list $\ell : \mathsf{List}(N)$ (using the name sort N to represent variables in λ-terms). Then the operation for forming λ-abstractions could be given type $\Pi(\ell : \mathsf{List}(N))\,(\text{И}[n:N]\mathsf{Term}(\mathsf{cons}\,n\,\ell)) \to \mathsf{Term}(\ell)$.

The notation for name abstraction types, $\text{И}[n : N]T$, is chosen to suggest its Curry-Howard relationship with the freshness quantifier of nominal logic [16, Sect. 3.2]; see Remark 3.7. Simultaneously it formalizes a dependent version of the nominal sets notion of name abstraction [16, Chapter 4]; see Sect. 3.6. In particular the elimination rule for name abstractions uses a special form of application, called *concretion* and written $t \,@\, n$. We know from the theory of nominal sets that to be meaningful such a concretion requires the name n to be fresh for the abstraction to be concreted, $t : \text{И}[n' : N]T$; and since types can depend upon names, we should also require n to be fresh for the type T. So the hypotheses of the И-elimination rule will be definitional freshness judgements $\Gamma[n' : N] \vdash (n : N)\,\#T$ and $\Gamma \vdash (n : N)\,\# t : \text{И}[n' : N]T$. What should be the type of the concretion $t \,@\, n$ in its conclusion? For dependent functions, Π-elimination involves making a substitution to get the result type of a function application:

$$\frac{\Gamma\vdash t:\Pi(x':T')T \qquad \Gamma\vdash t':T'}{\Gamma\vdash t\,t':T(t'/x')}\ (\Pi\text{-ELIM})$$

We cannot do the same for concretion and take its result type to be $T(n/n')$, because as we mentioned at the end of Sect. 2.1, validity of FreshMLTT judgements is not preserved in general by name-substitutions, only by name-permutations.[8] This

[8] Cheney's DNTT [3] does use name-substitution (see Fig. 9 in that paper), but constrains concreting names n to be meta-theoretically fresh for T, so that $T(n/n')$ is equal to the result of permuting n and n' in T. Definitional freshness is a weaker and hence more expressive constraint on concreting names.

suggest that we use the result type $(n\ \ n') \cdot T$ and the following elimination rule:

$$\frac{\Gamma[n' : N] \vdash (n : N) \# T \qquad \Gamma \vdash (n : N) \# t : \text{И}[n' : N]T}{\Gamma[n' : N] \vdash t @ n : (n\ \ n') \cdot T} \tag{2}$$

Note that the context in the conclusion has to be $\Gamma[n' : N]$ rather than just Γ, because the bound name n' in $\text{И}[n' : N]T$ becomes a free name in $(n\ \ n') \cdot T$. However, from the hypotheses of (2) (together with properties of name-swapping up to definitional equality) one can deduce $\Gamma[n' : N] \vdash (n' : N) \# (n\ \ n') \cdot T$ and (hence) $\Gamma[n'\ n'' : N] \vdash (n\ \ n') \cdot T = (n\ \ n'') \cdot T$. In other words the result type $(n\ \ n') \cdot T$ of the concretion in (2) is independent up to definitional equality of the choice of bound name n' in $\text{И}[n' : N]T$. It is important, from the point of view of expressivity of the system, to capture this fact linguistically. We can do so by using a form of local scoping for names in expressions, which we write as $\nu[n : N]\, e$. Such expressions can be introduced by the rules

$$\frac{\Gamma[n : N] \vdash (n : N) \# T}{\Gamma \vdash \nu[n : N]\, T} \text{ (LOCAL-TYP)} \qquad \frac{\Gamma[n : N] \vdash (n : N) \# t : T}{\Gamma \vdash \nu[n : N]\, t : \nu[n : N]\, T} \text{ (LOCAL-TERM)}$$

and manipulated with the computation rules

$$\frac{\Gamma[n : N] \vdash (n : N) \# T}{\Gamma[n : N] \vdash \nu[n : N]\, T = T} \text{ (LOCAL-TYP-COMP)}$$

$$\frac{\Gamma[n : N] \vdash (n : N) \# t : T}{\Gamma[n : N] \vdash \nu[n : N]\, t = t : T} \text{ (LOCAL-TERM-COMP)}$$

We will see later (Proposition 3.5 and Theorem 4.2) that these rules have a sound interpretation in terms of nominal sets. They formalize the form of locally fresh name commonly used in syntax-manipulating constructions, where the meaning of a construction depending upon some fresh names is (provably) independent of which particular fresh names are chosen; see [16, Sect. 3.3]. When the constructions involve first-class functions, it is important to have a linguistic form for this kind of name scoping, since an expression like $\lambda(x : T)\nu[n : N]\, t$ is not in general definitionally equal to $\nu[n : N]\, \lambda(x : T)t$ and so the local scoping cannot float to the top-level and become implicit in the context.

Using locally scoped names, we strengthen (2) to get FreshMLTT's rule for И-elimination:

$$\frac{\Gamma[n' : N] \vdash (n : N) \# T \qquad \Gamma \vdash (n : N) \# t : \text{И}[n' : N]T}{\Gamma \vdash t @ n : \nu[n' : N]\, (n\ \ n') \cdot T} \text{ (И-ELIM)}$$

$$\frac{\Gamma \vdash n : N \qquad \Gamma \vdash T}{\Gamma \vdash \langle\!\langle n : N \rangle\!\rangle T} \text{ (ABS-FORM)} \qquad \frac{\Gamma \vdash n : N \qquad \Gamma \vdash t : T}{\Gamma \vdash \langle n : N \rangle t : \langle\!\langle n : N \rangle\!\rangle T} \text{ (ABS-INTRO)}$$

$$\frac{\Gamma \vdash (n : N) \# t : \langle\!\langle n' : N \rangle\!\rangle T}{\Gamma \vdash t @ n : (n\ \ n') \cdot T} \text{ (ABS-ELIM)}$$

$$\frac{\Gamma \vdash n : N \qquad \Gamma \vdash (n' : N) \# t : T}{\Gamma \vdash (\langle n : N \rangle t) @ n' = (n\ \ n') \cdot t : (n\ \ n') \cdot T} \text{ (ABS-COMP)}$$

$$\frac{\Gamma \vdash (n : N) \# t : \langle\!\langle n : N \rangle\!\rangle T}{\Gamma \vdash \langle n : N \rangle (t @ n) = t : \langle\!\langle n : N \rangle\!\rangle T} \text{ (ABS-UNIQ)}$$

Fig. 3. Admissible rules for non-binding name abstraction

The associated computation and uniqueness (η) rules are:

$$\frac{\Gamma[n' : N] \vdash (n : N) \# t : T \qquad n \neq n'}{\Gamma \vdash (\alpha[n' : N]t) @ n = \nu[n' : N]\,(n\ \ n') \cdot t : \nu[n' : N]\,(n\ \ n') \cdot T} \text{ (И-COMP)}$$

$$\frac{\Gamma \vdash t : И[n : N]T \qquad n \notin \mathrm{fn}(t)}{\Gamma \vdash t = \alpha[n : N](t @ n) : И[n : N]T} \text{ (И-UNIQ)}$$

We will see in Sect. 4 that they are sound for the nominal set semantics developed in Sect. 3.

2.4 Non-binding name abstraction

FreshML [21] uses a non-binding form of name abstraction which is definable in FreshMLTT as follows:

$$\langle\!\langle n : N \rangle\!\rangle T \triangleq И[n' : N](n\ \ n') \cdot T \tag{3}$$

$$\langle n : N \rangle t \triangleq \alpha[n' : N](n\ \ n') \cdot t \tag{4}$$

(where $n' \notin \mathrm{fn}(n, T, t)$). Note that n occurs free in $\langle\!\langle n : N \rangle\!\rangle T$ and $\langle n : N \rangle T$, whereas it is bound in $И[n : N]T$ and $\alpha[n : N]t$. Admissible rules for this form of name abstraction are shown in Fig. 3.

If $\Gamma[n\ n' : N] \vdash t : T$, then it is a consequence of (FRESH-HYP-TYP) and the conversion properties of the swapping operation $(n\ \ n') \cdot \_$ that $\Gamma[n\ n' : N] \vdash (n\ \ n') \cdot T = T'$ and $\Gamma[n\ n' : N] \vdash (n\ \ n') \cdot t = t' : T'$, where t' and T' are the expressions obtained from t

and T by transposing occurrences of n and n'. It follows that

$$\frac{\Gamma[n:N] \vdash T}{\Gamma \vdash \mathsf{И}[n:N]T = \nu[n:N]\,\langle\!\langle n:N\rangle\!\rangle T} \tag{5}$$

$$\frac{\Gamma[n:N] \vdash t : T}{\Gamma \vdash \alpha[n:N]t = \nu[n:N]\,\langle n:N\rangle t : \mathsf{И}[n:N]T} \tag{6}$$

are admissible rules. Conversely, one could take the non-binding form of name abstraction as primitive, with rules as in Fig. 3, and then define the binding form from it using locally scoped names as in (5) and (6).

Non-binding name abstractions in the FreshMLTT meta-language are used when one has to refer to an object-level bound name in more than one textual location. Here is an example.

Example 2.2 As usual we write $T \to T'$ for the non-dependent function type, that is, for $\Pi(x:T)T'$ when $x \notin \mathrm{fv}(T')$. For non-dependent name abstractions we write

$$\langle\!\langle N\rangle\!\rangle T \;\triangleq\; \mathsf{И}[n:N]T \qquad \text{where } n \notin \mathrm{fn}(T) \tag{7}$$

(which is definitionally equal to the non-binding abstraction $\langle\!\langle n:N\rangle\!\rangle T$ when $n \notin \mathrm{fn}(T)$). In the nominal sets semantics given below, the type $\langle\!\langle N\rangle\!\rangle T$ is modelled by a nominal set of name abstractions [16, Chapter 4]. These are known to commute with exponentiation up to isomorphism in the category of nominal sets (see [16, Proposition 4.14]); and indeed we can express an isomorphism $\langle\!\langle N\rangle\!\rangle(T \to T') \cong \langle\!\langle N\rangle\!\rangle T \to \langle\!\langle N\rangle\!\rangle T'$ within FreshMLTT. In fact we can give a dependently typed generalisation of this isomorphism

$$\mathsf{И}[n:N]\,\Pi(x:T)\,T' \;\cong\; \Pi(y:\mathsf{И}[n:N]T)\,\mathsf{И}[n:N]\,T'(y \mathbin{@} n/x) \tag{8}$$

as follows. Define

$$\begin{aligned} T_1 &\triangleq \mathsf{И}[n:N]\,\Pi(x:T)T' \\ T_2 &\triangleq \Pi(y:\mathsf{И}[n:N]T)\mathsf{И}[n:N]T'(y \mathbin{@} n/x) \\ i &\triangleq \lambda(z:T_1)\,\lambda(y:\mathsf{И}[n:N]T)\,\alpha[n:N]\;(z \mathbin{@} n)(y \mathbin{@} n) \\ j &\triangleq \lambda(f:T_2)\,\alpha[n:N]\,\lambda(x:T)\;f(\langle n:N\rangle x) \mathbin{@} n \end{aligned}$$

Using the derived rules in Fig. 3 one can show that if

$$\begin{aligned} \Gamma[n:N] &\vdash T \\ \Gamma[n:N](x:T) &\vdash T' \end{aligned}$$

are provable in FreshMLTT, then so are

$$\Gamma \vdash i : T_1 \to T_2$$
$$\Gamma \vdash j : T_2 \to T_1$$
$$\Gamma(z : T_1) \vdash j(i\,z) = z : T_1$$
$$\Gamma(f : T_2)) \vdash i(j\,f) = f : T_2$$

Note that in the subexpression $f(\langle n : N\rangle x)$ @ n of j we use the non-binding name abstraction $\langle n : N\rangle x$, rather than $\alpha[n : N]x$ (which is equal up to α-equivalence to $\alpha[n' : N]x$ for any n'), because after applying the function f we have to concrete at the same name n in order for the typing to work out.

3 Families of Nominal Sets

We will give a semantics to FreshMLTT using nominal sets with sorted atoms. Thus we assume there is a fixed set $\mathbb{A}$ of *atoms*, partitioned into countably infinitely many subsets $\mathbb{A} = \biguplus_{N \in Nsort} \mathbb{A}_N$, indexed by the sorts of name $N \in Nsort$ in FreshMLTT. Each subset $\mathbb{A}_N$ is countably infinite. We write $\mathsf{a}, \mathsf{b}, \ldots$ for typical elements of $\mathbb{A}$.

Let G be the group of permutations π of $\mathbb{A}$ that respect sorts ($\pi\,\mathsf{a} \in \mathbb{A}_N$, if $\mathsf{a} \in \mathbb{A}_N$) and are finite (in the sense that $\pi\,\mathsf{a} = \mathsf{a}$ for all but finitely many $\mathsf{a} \in \mathbb{A}$). A *nominal set* is a set X equipped with a G-action with respect to which every element has a finite support. This means that for each $x \in X$ there is a finite subset $A \subseteq_{\text{fin}} \mathbb{A}$ satisfying $(\forall \pi \in G_A)\ \pi \cdot x = x$, where $G_A \triangleq \{\pi \in G \mid (\forall \mathsf{a} \in A)\ \pi\,\mathsf{a} = \mathsf{a}\}$. Nominal sets are the objects of a category that we will denote by **Nom** and whose morphisms are equivariant functions ($f(\pi \cdot x) = \pi \cdot (f\,x)$), with composition and identities as in the category of sets. We refer the reader to [16] for an introduction to nominal sets and in particular to Sect. 4.7 of that book for the many-sorted version we use here.

3.1 **Nom** *as a category with families*

To model FreshMLTT we will endow **Nom** with the structure of a *category with families* (CwF) [8]. In general a CwF is specified by a category $\mathbf{C}$ with a terminal object 1, together with the following structure:

- For each object $X \in \mathbf{C}$, a collection $\mathbf{C}(X)$, whose elements are called *families* over X.
- For each object $X \in \mathbf{C}$ and family $E \in \mathbf{C}(X)$, a collection $\mathbf{C}(X \vdash E)$ of *elements* of the family E over X.
- Operations for re-indexing families and elements along morphisms in $\mathbf{C}$

$$\frac{E \in \mathbf{C}(X) \qquad f \in \mathbf{C}(Y, X)}{E[f] \in \mathbf{C}(Y)} \qquad \frac{e \in \mathbf{C}(X \vdash E) \qquad f \in \mathbf{C}(Y, X)}{e[f] \in \mathbf{C}(Y \vdash E[f])}$$

satisfying

$$E[\mathrm{id}_X] = E \qquad (A \in \mathbf{C}(X)) \tag{9}$$
$$E[f \circ g] = E[f][g] \qquad (E \in \mathbf{C}(X), f \in \mathbf{C}(Y,X), g \in \mathbf{C}(Z,Y)) \tag{10}$$
$$e[\mathrm{id}_X] = e \qquad (e \in \mathbf{C}(X \vdash E) \tag{11}$$
$$e[f \circ g] = e[f][g] \qquad (e \in \mathbf{C}(X \vdash E), f \in \mathbf{C}(Y,X), g \in \mathbf{C}(Z,Y)) \tag{12}$$

- For each family $E \in \mathbf{C}(X)$, a *comprehension* object $X.E \in \mathbf{C}$ equipped with a projection morphism $\mathrm{p} \in \mathbf{C}(X.E, X)$, a *generic element* $\mathrm{v} \in \mathbf{C}(X.E \vdash E[\mathrm{p}])$ and a *pairing operation*

$$\frac{f \in \mathbf{C}(Y,X) \qquad E \in \mathbf{C}(X) \qquad e \in \mathbf{C}(Y \vdash E[f])}{\langle f\,,e\rangle \in \mathbf{C}(Y, X.E)}$$

satisfying

$$\mathrm{p} \circ \langle f\,,e\rangle = f \tag{13}$$
$$\mathrm{v}[\langle f\,,e\rangle] = e \tag{14}$$
$$\langle f\,,e\rangle \circ g = \langle f \circ g\,, e[g]\rangle \tag{15}$$
$$\langle \mathrm{p}\,,\mathrm{v}\rangle = \mathrm{id}_{X.E} \tag{16}$$

Definition 3.1 We make the category **Nom** of nominal sets and equivariant functions into a CwF as follows:

- The collection $\mathbf{Nom}(X)$ of *families of nominal sets* E over a nominal set $X \in \mathbf{Nom}$ consists of X-indexed families of sets $(E_x \mid x \in X)$ equipped with a *dependent G-action*

$$act_E \in \textstyle\prod_{x\in X} \prod_{\pi\in G} E_x \to E_{\pi\cdot x} \tag{17}$$

satisfying a finite support property given below. We will write $\pi \cdot e$ for the application of act_E to $x \in X, \pi \in G, e \in E_x$, leaving E and x as implicit arguments. To qualify as an action (17) has to satisfy for all $x \in X$, $\pi, \pi' \in G$ and $e \in E_x$

$$\begin{array}{ccc} \pi' \cdot (\pi \cdot e) & = & \pi'\pi \cdot e \\ \cap & & \cap \\ E_{\pi'\cdot(\pi\cdot x)} & = & E_{\pi'\pi\cdot x} \end{array} \qquad \text{and} \qquad \begin{array}{ccc} \iota \cdot e & = & e \\ \cap & & \cap \\ E_{\iota\cdot x} & = & E_x \end{array} \tag{18}$$

(where ι denotes the identity permutation). In addition act is required to satisfy the following finite support property: for every $x \in X$ and $e \in E_x$ there is a finite set $A \subseteq_{\mathrm{fin}} \mathbb{A}$ of atoms satisfying

$$(\forall \pi \in G_A)\ \pi \cdot x = x \ \wedge\ \pi \cdot e = e$$

(so that in particular A supports x in X and hence $E_{\pi\cdot x} = E_x$ for any $\pi \in G_A$). In this case we will say that A is a *finite support for e dependent upon x*.

- The collection $\mathbf{Nom}(X \vdash E)$ of elements of a family $E \in \mathbf{Nom}(X)$ consists of those dependent functions $e \in \prod_{x\in X} E_x$ that are *dependently equivariant*, in the sense that $\pi \cdot (e\, x) = e(\pi \cdot x) \in E_{\pi\cdot x}$, for all $x \in X$ and $\pi \in G$.
- The re-indexing of $E \in \mathbf{Nom}(X)$ along $f \in \mathbf{Nom}(Y, X)$ is the Y-indexed family of sets $E[f] \triangleq (E_{f\,y} \mid y \in Y)$ with dependently-typed G-action inherited from E: if $e \in E[f]_y = E_{f\,y}$, then we get $\pi \cdot e \in E_{\pi\cdot(f\,y)} = E_{f(\pi\cdot y)} = E[f]_{\pi\cdot y}$ (using the fact f, being a morphisms in **Nom**, is equivariant). Similarly, the re-indexing of $e \in \mathbf{Nom}(X \vdash E)$ along $f \in \mathbf{Nom}(Y, X)$ is $e[f] \triangleq \lambda(y \in Y) \to e(f\,y)$, which is in $\mathbf{Nom}(Y \vdash E[f])$, because e is dependently equivariant and f is equivariant.
- For each $E \in \mathbf{Nom}(X)$, the comprehension object $X.E \in \mathbf{Nom}$ is the nominal set given by the disjoint union of sets $\sum_{x\in X} E_x$ equipped with the G-action mapping $(x, e) \in \sum_{x\in X} E_x$ to $\pi \cdot (x, e) = (\pi \cdot x, \pi \cdot e)$, given by the G-action of X in the first component and the dependent G-action of E in the second component. Note that by definition of $\mathbf{Nom}(X)$, every $(x, e) \in \sum_{x\in X} E_x$ is finitely supported with respect to this G-action and hence $X.E$ is indeed a nominal set.

  The projection morphism $\mathrm{p} \in \mathbf{Nom}(X.E, X)$ is given by first projection: $\mathrm{p}(x, e) \triangleq x$. The generic element $\mathrm{v} \in \mathbf{Nom}(X.E \vdash E[\mathrm{p}])$ is given by second projection: $\mathrm{v}(x, e) \triangleq e \in E_x = E[\mathrm{p}]_{(x,e)}$. Given $E \in \mathbf{Nom}(X)$, the pairing of $f \in \mathbf{Nom}(Y, X)$ and $e \in \mathbf{Nom}(Y \vdash E[f])$ is the equivariant function $\langle f, e\rangle \in \mathbf{Nom}(Y, X.E)$ given by mapping each $y \in Y$ to $\langle f, e\rangle\, y \triangleq (f\, y, e\, y) \in X.E$.

It is easy to check that these definitions satisfy (9)–(16) and so make **Nom** into a CwF.

Remark 3.2 For each object $X \in \mathbf{C}$ of a CwF, one can make $\mathbf{C}(X)$ into a category by taking, for each $E, E' \in \mathbf{C}(X)$, the set of morphisms $\mathbf{C}(X)(E, E')$ to be $\mathbf{C}(X.E \vdash E'[\mathrm{p}])$ with identities given by generic elements and composition given by $e' \circ e \triangleq e'[\langle \mathrm{p}\,, e\rangle]$. Then the mapping $E \in \mathbf{C}(X) \mapsto \mathrm{p} \in \mathbf{C}(X.E, X)$ extends to a full and faithful functor to the slice category

$$\begin{array}{c} \mathbf{C}(X) \to \mathbf{C}/X \\ E \xrightarrow{e} E' \mapsto X.E \xrightarrow{\langle \mathrm{p}, e\rangle} X.E' \\ \qquad\qquad\quad \searrow^{\mathrm{p}} \quad \swarrow_{\mathrm{p}} \\ \qquad\qquad\quad X \end{array} \tag{19}$$

The re-indexing operations are mapped to pullback functors between slices, since for each $E \in \mathbf{C}(X)$ and $f \in \mathbf{C}(Y, X)$

$$\begin{array}{ccc} Y.E[f] & \xrightarrow{\langle f\circ \mathrm{p}, \mathrm{v}\rangle} & X.E \\ {\scriptstyle \mathrm{p}}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathrm{p}} \\ Y & \xrightarrow[f]{} & X \end{array} \tag{20}$$

is a pullback in $\mathbf{C}$; see [11, Proposition 3.9]. When $\mathbf{C} = \mathbf{Nom}$, the functors (19) are not only full and faithful, but also essentially surjective and hence equivalences. This

$$\frac{E \in \mathbf{C}(X) \qquad F \in \mathbf{C}(X.E)}{\Pi\, E\, F \in \mathbf{C}(X)} \qquad\qquad \frac{f \in \mathbf{C}(X.E \vdash F)}{\mathrm{lam}\, f \in \mathbf{C}(X \vdash \Pi\, E\, F)}$$

$$\frac{f \in \mathbf{C}(X \vdash \Pi\, E\, F) \qquad e \in \mathbf{C}(X \vdash E)}{\mathrm{app}\, f\, e \in \mathbf{C}(X \vdash F[\langle \mathrm{id}_X\, , e\rangle])}$$

$$\begin{aligned}
(\Pi\, E\, F)[g] &= \Pi(E[g])(F[\langle g \circ \mathrm{p}\, , \mathrm{v}\rangle]) \\
(\mathrm{lam}\, f)[g] &= \mathrm{lam}\, f[\langle g \circ \mathrm{p}\, , \mathrm{v}\rangle] \\
(\mathrm{app}\, f\, e)[g] &= \mathrm{app}\, (f[g])\, (e[g]) \\
\mathrm{app}\, (\mathrm{lam}\, f)\, e &= f[\langle \mathrm{id}_X\, , e\rangle] \\
\mathrm{lam}(\mathrm{app}\, (f[\mathrm{p}])\, \mathrm{v}) &= f
\end{aligned}$$

Fig. 4. Π-types in a CwF

is because, given $p : E \to X$ in $\mathbf{Nom}/X$, the X-indexed family of sets $(p^{-1}\{x\} \mid x \in X)$ inherits a dependent G-action from the G-action of E (since p is equivariant); and for each $x \in X$, if $A \subseteq_{\mathrm{fin}} \mathbb{A}$ supports e in E and $p\, e = x$, then it is a support for $e \in p^{-1}\{x\}$ dependent upon x, in the sense of Definition 3.1. So $(p^{-1}\{x\} \mid x \in X)$ is an object of $\mathbf{Nom}(X)$. It is sent by the functor (19) to

$$\pi_1 : \textstyle\sum_{x \in X} p^{-1}\{x\} = \{(x, e) \mid p\, e = x\} \to X$$

which is isomorphic in $\mathbf{Nom}/X$ to $p : E \to X$ via the second projection function $\pi_2 : (x, e) \mapsto e$, whose inverse is $e \mapsto (p\, e, e)$.

3.2 Π-*types in* **Nom**

The contexts, types-in-context, terms-in-context and term-substitutions of Martin-Löf Type Theory are interpreted in a CwF by its objects, families, elements and morphisms respectively; see [11, Sect. 3.5]. CwFs provide an essentially algebraic formulation of Type Theory syntax in nameless (de Bruijn index) style and consequently one can translate each type-forming construct to an equivalent structure within CwFs. For example, the extra structure $(\Pi, \mathrm{lam}, \mathrm{app})$ corresponding to Π-types is given in Fig. 4. Since **Nom** is a topos and hence is in particular locally cartesian closed, it follows from Remark 3.2 that as a CwF **Nom** has this structure. We can describe $(\Pi, \mathrm{lam}, \mathrm{app})$ in this case as follows:

- Π: given $E \in \mathbf{Nom}(X)$ and $F \in \mathbf{Nom}(X.E)$, first note that we get a dependent G-action on the X-indexed family of sets $(\prod_{e \in E_x} F_{(x,e)} \mid x \in X)$ by mapping $f \in \prod_{e \in E_x} F_{(x,e)}$ to $\pi \cdot f \triangleq \lambda(e \in E_{\pi \cdot x}) \to \pi \cdot (f(\pi^{-1} \cdot e))$. To get a family $\Pi\, E\, F \in \mathbf{Nom}(X)$, for each $x \in X$ we take $(\Pi\, E\, F)_x$ to be the subset of $\prod_{e \in E_x} F_{(x,e)}$ consisting of those f that have a finite support dependent upon x with respect to the above action.

- lam: if $f \in \mathbf{Nom}(X.E \vdash F)$, then for each $x \in X$ one can check that

$$\mathrm{lam}\, f\, x \triangleq \lambda(e \in E_x) \to f(x,e) \in \textstyle\prod_{e \in E_x} F_{(x,e)}$$

  is supported (dependently upon x) by any finite subset $A \subseteq_{\mathrm{fin}} \mathbb{A}$ supporting x in X; hence $\mathrm{lam}\, f\, x \in (\Pi\, E\, F)_x$. Furthermore $x \mapsto \mathrm{lam}\, f\, x$ is dependently equivariant (because f is) and hence $\mathrm{lam}\, f \in \mathbf{Nom}(X \vdash \Pi\, E\, F)$.
- app : if $f \in \mathbf{Nom}(X \vdash \Pi\, E\, F)$ and $e \in \mathbf{Nom}(X \vdash E)$, then for each $x \in X$ we have $f\, x \in (\Pi\, E\, F)_x \subseteq \prod_{e \in E_x} F_{(x,e)}$ and $e\, x \in E_x$; hence

$$\mathrm{app}\, f\, e\, x \triangleq (f\, x)(e\, x) \in F_{(x, e\, x)} = F[\langle \mathrm{id}\, , e \rangle]_x$$

  and one can check that $x \mapsto \mathrm{app}\, f\, e\, x$ is dependently equivariant, because f and e are. So we get $\mathrm{app}\, f\, e \in \mathbf{Nom}(X \vdash F[\langle \mathrm{id}\, , e \rangle])$.

It is easy to see that these operations satisfy the properties in Fig. 4.

3.3 Name objects in **Nom**

For each sort of names $N \in \mathit{Nsort}$, the set $\mathbb{A}_N$ of atoms of that sort becomes a nominal set once we endow it with the G-action given by function application, $\pi \cdot a \triangleq \pi\, a$, with respect to which each $a \in \mathbb{A}_N$ is supported by $\{a\}$.

From Remark 3.2 we have that the category $\mathbf{Nom}(1)$ of families over the terminal object 1 is equivalent to **Nom**. For each $X \in \mathbf{Nom}$ we will not make a notational distinction between a nominal set Y and the corresponding family in $\mathbf{Nom}(X)$ which is constant with value Y. In particular, for each sort of names $N \in \mathit{Nsort}$, we just write $\mathbb{A}_N \in \mathbf{Nom}(X)$ for an object of names regarded as a constant family over X. The elements $a \in \mathbf{Nom}(X \vdash \mathbb{A}_N)$ of this family are just the equivariant functions $a \in \mathbf{Nom}(X, \mathbb{A}_N)$.

To interpret branching on name equality we will use the following operation in the CwF **Nom**:

$$\frac{a, b \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad e, f \in \mathbf{Nom}(X \vdash E)}{\mathrm{ifeq}(a, b, e, f) \in \mathbf{Nom}(X \vdash E)} \tag{21}$$

where

$$\mathrm{ifeq}(a, b, e, f)\, x \triangleq \begin{cases} e\, x & \text{if } a\, x = b\, x \\ f\, x & \text{otherwise} \end{cases} \qquad (x \in X) \tag{22}$$

3.4 Swapping names in context

If $\mathsf{a}, \mathsf{b} \in \mathbb{A}_N$ are atoms of the same sort N, as usual we write $(\mathsf{a}\ \ \mathsf{b})$ for the permutation in G that interchanges a with b, leaving all other atoms fixed. We lift the action $x \mapsto (\mathsf{a}\ \ \mathsf{b}) \cdot x$ from elements of nominal sets to families and their elements in the CwF **Nom** as follows

The operation

$$\frac{a, b \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad E \in \mathbf{Nom}(X)}{(a\ \ b) \cdot E \in \mathbf{Nom}(X)} \tag{23}$$

is defined by

$$((a\ \ b) \cdot E)_x \triangleq E_{(b\,x\ \ a\,x)\cdot x} \qquad (x \in X)$$

with dependent G-action

$$\begin{aligned} act_{(a\ \ b)\cdot E}\, x\, \pi\, e &\triangleq act_E\, ((b\,x\ \ a\,x) \cdot x)\, \pi\, e \qquad (x \in X, \pi \in G, e \in ((a\ \ b) \cdot E)_x) \\ &\in E_{\pi\cdot(b\,x\ \ a\,x)\cdot x} \\ &= E_{(b(\pi\cdot x)\ \ a(\pi\cdot x))\cdot\pi\cdot x} \qquad \text{(using equivariance of } a \text{ and } b) \\ &= ((a\ \ b) \cdot E)_{\pi\cdot x} \end{aligned}$$

In other words, the action of π on $e \in ((a\ \ b)\cdot E)_x$ is the action of π on $e \in E_{(b\,x\ \ a\,x)\cdot x}$ given by the family E. [9]

Similarly, the operation

$$\frac{a, b \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad e \in \mathbf{Nom}(X \vdash E)}{(a\ \ b) \cdot e \in \mathbf{Nom}(X \vdash (a\ \ b) \cdot E)} \tag{24}$$

is defined by

$$\begin{aligned} ((a\ \ b) \cdot e)\, x &\triangleq e((b\,x\ \ a\,x) \cdot x) \qquad (x \in X) \\ &\in E_{(b\,x\ \ a\,x)\cdot x} = ((a\ \ b) \cdot E)_x \end{aligned}$$

which is dependently equivariant with respect to the above dependent G-action, since $((a\ \ b)\cdot e)\,(\pi\cdot x) \triangleq e((b(\pi\cdot x)\ \ a(\pi\cdot x))\cdot(\pi\cdot x)) = \pi\cdot e((b\,x\ \ a\,x)\cdot x) = \pi\cdot((a\ \ b)\cdot e)\,x$, using equivariance of a, b and dependent equivariance of e.

3.5 Freshness in context

If $X \in \mathbf{Nom}$, $x \in X$ and $\mathsf{a} \in \mathbb{A}$, we write $\mathsf{a} \# x$ for the usual nominal sets *freshness relation*: by definition it means that there is a finite subset $A \subseteq_{\text{fin}} \mathbb{A}$ supporting x in X with $\mathsf{a} \notin A$; see [16, Sect. 3.1]. If $E \in \mathbf{Nom}(X)$, then the freshness relation for the nominal set $X.E$ corresponds to a dependent version of freshness: if $x \in X$ and $e \in E_x$, then by Definition 3.1, there is some finite subset $A \subseteq_{\text{fin}} \mathbb{A} - \{\mathsf{a}\}$ supporting e dependent upon x iff $\mathsf{a} \# (x, e)$ holds for the nominal set $X.E$. In this case it is tempting to write '$\mathsf{a} \# e$', but will avoid doing so, because it is easy to forget that $\mathsf{a} \# x$ is also required for this to make sense.

[9] We write $(b\,x\ \ a\,x)\cdot x$ in the above, rather than $(a\,x\ \ b\,x)\cdot x$, to indicate that when using equivariant functions valued in general permutations rather than just transpositions, we should define $(\pi \cdot E)_x$ to be $E_{(\pi\,x)^{-1}\cdot x}$. Of course $(b\,x\ \ a\,x)\cdot x = (a\,x\ \ b\,x)\cdot x$, since transpositions are self-inverse.

As well as freshness for elements of families, we will need a notion of freshness for families themselves. Given $E \in \mathbf{Nom}(X)$ and $x \in X$, it makes no sense to write '$\mathsf{a} \# E_x$' because, given the way we have defined families of nominal sets, E_x is not an element of a nominal set. Instead we use the fact that the freshness relation can be characterised equationally: $\mathsf{a} \# x$ holds iff $(\mathsf{a}\ \ \mathsf{b}) \cdot x = x$ holds for some (or indeed, any) fresh b. In the CwF **Nom** we can replace 'fresh b' with use of the *separated product* [16, Sect. 3.4]:

$$X * \mathbb{A}_N \triangleq \{(x, \mathsf{a}) \in X \times \mathbb{A}_N \mid \mathsf{a} \# x\} \tag{25}$$

This is a nominal subset of the product: the G-action on $X * \mathbb{A}_N$ is well-defined by $\pi \cdot (x, \mathsf{a}) = (\pi \cdot x, \pi\, \mathsf{a})$ (since the freshness relation is equivariant); and with respect to this action (x, a) is finitely supported by $A \cup \{\mathsf{a}\}$, if $A \subseteq_{\mathrm{fin}} \mathbb{A}$ supports x in X. This separated product will be used to model the extension of typing contexts with a fresh name. First and second projection yield

$$\mathrm{p} \in \mathbf{Nom}(X * \mathbb{A}_N, X) \qquad \mathrm{p}(x, \mathsf{a}) \triangleq x \tag{26}$$

$$\nu \in \mathbf{Nom}(X * \mathbb{A}_N \vdash \mathbb{A}_N) \qquad \nu(x, \mathsf{a}) \triangleq \mathsf{a} \tag{27}$$

The first will be used to model weakening a judgement from a typing context to one extended with a fresh name and the second will model the fresh name itself.

Definition 3.3 Given $a \in \mathbf{Nom}(X \vdash \mathbb{A}_N)$, using the operations in (26) and (27) we get $a[\mathrm{p}], \nu \in \mathbf{Nom}(X * \mathbb{A}_N \vdash \mathbb{A}_N)$. Then for any family $E \in \mathbf{Nom}(X)$ and any element $e \in \mathbf{Nom}(X \vdash E)$, using the swapping operations from Sect. 3.4 we define

$$a \#_X E \triangleq (a[\mathrm{p}]\ \ \nu) \cdot E[\mathrm{p}] = E[\mathrm{p}] \in \mathbf{Nom}(X * \mathbb{A}_N) \tag{28}$$

$$a \#_X e \triangleq a \#_X E \wedge (a[\mathrm{p}]\ \ \nu) \cdot e[\mathrm{p}] = e[\mathrm{p}] \in \mathbf{Nom}(X * \mathbb{A}_N \vdash E[\mathrm{p}]) \tag{29}$$

Note that the equality of families of nominal sets in (28) means not only that for all $(x, \mathsf{b}) \in X * \mathbb{A}_N$ the sets $E_{(a\ x\ \mathsf{b}) \cdot x}$ and E_x are equal, but also that the dependent G-actions on these two families of sets are equal.

The following result follows easily from Definition 3.3:

Proposition 3.4 *Suppose* $a, a' \in \mathbf{Nom}(X \vdash \mathbb{A}_N)$ *and* $e \in \mathbf{Nom}(X \vdash E)$. *Then*

$$a \#_X E \Rightarrow a' \#_X (a\ \ a') \cdot E \tag{30}$$

$$a \#_X e \Rightarrow a' \#_X (a\ \ a') \cdot e \tag{31}$$

$$\nu \#_{X * \mathbb{A}_N} e[\mathrm{p}] \tag{32}$$

□

Proposition 3.5 (name restriction) *There are operations*

$$\frac{E \in \mathbf{Nom}(X * \mathbb{A}_N) \qquad \nu \#_{X*\mathbb{A}_N} E}{\operatorname{res} E \in \mathbf{Nom}(X)} \tag{33}$$

$$\frac{e \in \mathbf{Nom}(X * \mathbb{A}_N \vdash E) \qquad \nu \#_{X*\mathbb{A}_N} e}{\operatorname{res} e \in \mathbf{Nom}(X \vdash \operatorname{res} E)} \tag{34}$$

satisfying

$$(\operatorname{res} E)[\mathrm{p}] = E \in \mathbf{Nom}(X * \mathbb{A}_N) \tag{35}$$

$$(\operatorname{res} e)[\mathrm{p}] = e \in \mathbf{Nom}(X * \mathbb{A}_N \vdash E) \tag{36}$$

We call $\operatorname{res} E$ *a* name restricted *family and* $\operatorname{res} e$ *a* name restricted *element (of a name restricted family).*

Proof. From definition (28) we have that $\nu \#_{X*\mathbb{A}_N} E$ implies that $E_{(x,\mathsf{a})} = E_{(x,\mathsf{b})}$ for all $x \in X$ and $\mathsf{a}, \mathsf{b} \# x$ (and the dependent G-actions on each family are equal). So $\operatorname{res} E$ is well-defined by

$$(\operatorname{res} E)_x \triangleq E_{(x,\mathsf{a})} \text{ where } \mathsf{a} \# x \qquad (x \in X) \tag{37}$$

and clearly satisfies $(\operatorname{res} E)[\mathrm{p}] = E$. Similarly, if $\nu \#_{X*\mathbb{A}_N} e$, then definition (29) implies that $e(x, \mathsf{a}) = e(x, \mathsf{b}) \in E_{(x,\mathsf{b})} = E_{(x,\mathsf{a})}$ for all $x \in X$ and $\mathsf{a}, \mathsf{b} \# x$. So $\operatorname{res} e$ is well-defined by

$$(\operatorname{res} e)\, x \triangleq e(x, \mathsf{a}) \text{ where } \mathsf{a} \# x \qquad (x \in X) \tag{38}$$

and clearly satisfies $(\operatorname{res} e)[\mathrm{p}] = e$. □

The form of locally scoped name embodied by the above proposition might seem trivial, but it is exactly the kind that occurs in connection with concreting name abstractions, as we see next.

3.6 Dependent name abstraction

We next describe structure in the CwF **Nom** that enables us to model dependently typed name abstraction, generalizing the usual notion of name abstraction for nominal sets [16, Chapter 4] and putting into the context of CwFs the fibred category construct Σ_N^* from [18, Sect. 3.3.2].

Definition 3.6 Given $E \in \mathbf{Nom}(X * \mathbb{A}_N)$, consider the X-indexed family of sets $И_N E$, where for each $x \in X$

$$(И_N E)_x \triangleq \{(\mathsf{a}, e) \mid \mathsf{a} \in \mathbb{A}_N \wedge \mathsf{a} \# x \wedge e \in E_{(x,\mathsf{a})}\} / \approx_x$$

is a quotient set. The equivalence relation $\approx_x$ relates (a, e) and (a', e') iff $(\mathsf{b}\ \ \mathsf{a}) \cdot e = (\mathsf{b}\ \ \mathsf{a}') \cdot e' \in E_{(x,\mathsf{b})}$ holds for some $\mathsf{b} \in \mathbb{A}_N$ with $\mathsf{b} \neq \mathsf{a}$, $\mathsf{b} \neq \mathsf{a}'$, $\mathsf{b} \# (x, e)$ and

$\mathsf{b} \# (x, e')$, or equivalently, for any such b. We will write the $\approx_x$-equivalence class of (a, e) as $\langle \mathsf{a} \rangle_x e$. We get a well-defined dependent G-action on $И_N E$ by defining $\pi \cdot \langle \mathsf{a} \rangle_x e \triangleq \langle \pi\, \mathsf{a} \rangle_{\pi \cdot x} (\pi \cdot e)$. If $A \subseteq_{\mathrm{fin}} \mathbb{A}$ supports $e \in E_{(x,\mathsf{a})}$ dependent upon (x, a), then it also supports $\langle \mathsf{a} \rangle_x e \in (И_N E)_x$. (In fact $A - \{\mathsf{a}\}$ supports $\langle \mathsf{a} \rangle_x e$, as the next lemma shows.) So we get an operation

$$\frac{E \in \mathbf{Nom}(X * \mathbb{A}_N)}{И_N E \in \mathbf{Nom}(X)} \tag{39}$$

that we call *dependent name abstraction* for families of nominal sets.

Remark 3.7 (Curry-Howard for И) If $X \in \mathbf{Nom}$ and $\varphi(\mathsf{a}, x)$ is a property of elements of $\mathbb{A}_N \times X$ that is equivariant (that is, $(\forall \pi \in G)\ \varphi(\mathsf{a}, x) \Rightarrow \varphi(\pi\, \mathsf{a}, \pi \cdot x)$), then the *freshness quantifier* $(И\mathsf{a} \in \mathbb{A}_N)\, \varphi(\mathsf{a}, x)$ means that $\varphi(\mathsf{a}, x)$ holds for some $\mathsf{a} \in \mathbb{A}_N$ with $\mathsf{a} \# x$, or equivalently, for any such a; see [16, Sect. 3.2]. There is a form of Curry-Howard correspondence between this quantifier and dependent name abstraction. For we can regard each family $E \in \mathbf{Nom}(X)$ as an equivariant property $\varphi_E(x)$ of elements $x \in E$, by defining $\varphi_E(x)$ to hold iff E_x is inhabited. Then $\varphi_{И_N E}(x)$ holds iff $(И\mathsf{a} \in \mathbb{A}_N)\, \varphi_E(x, \mathsf{a})$.

Lemma 3.8 *Suppose $E \in \mathbf{Nom}(X * \mathbb{A}_N)$ and $x \in X$.*

(i) *For all $\langle \mathsf{a} \rangle_x e \in (И_N E)_x$ and $\mathsf{a}' \in \mathbb{A}_N$, $\mathsf{a}' \# (x, \langle \mathsf{a} \rangle_x e) \in X.И_N E$ iff either $\mathsf{a}' = \mathsf{a}$, or $\mathsf{a}' \# ((x, \mathsf{a}), e) \in (X * \mathbb{A}_N).E$.*

(ii) *For each $f \in (И_N E)_x$ and $\mathsf{a} \in \mathbb{A}_N$ with $\mathsf{a} \# (x, f) \in X.И_N E$, there is a unique $f\, @_x\, \mathsf{a} \in E_{(x,\mathsf{a})}$, called the* concretion *of f at a, such that $f = \langle \mathsf{a} \rangle_x (f\, @_x\, \mathsf{a})$.*

Proof.

(i) The proof is similar to the proof of [16, Proposition 4.5].

(ii) Existence: by definition of $И_N E$, f is of the form $\langle \mathsf{a}' \rangle_x e$ for some $\mathsf{a}' \# x$ and $e \in E_{(x,\mathsf{a}')}$. If $\mathsf{a}' = \mathsf{a}$, we can take $f\, @_x\, \mathsf{a} = e$; otherwise, by part (i) we have $\mathsf{a} \# ((x, \mathsf{a}'), e)$ and hence $(\mathsf{a}\ \ \mathsf{a}') \cdot e \in E_{(x,\mathsf{a})}$ with $(\mathsf{a}', e) \approx_x (\mathsf{a}, (\mathsf{a}\ \ \mathsf{a}') \cdot e)$. Therefore $f = \langle \mathsf{a}' \rangle_x e = \langle \mathsf{a} \rangle_x (\mathsf{a}\ \ \mathsf{a}') \cdot e$ and we can take $f\, @_x\, \mathsf{a} = (\mathsf{a}\ \ \mathsf{a}') \cdot e$.

Uniqueness: if $\langle \mathsf{a} \rangle_x e = \langle \mathsf{a} \rangle_x e' \in (И_N E)_x$, then $(\mathsf{a}, e) \approx_x (\mathsf{a}, e')$ and hence for a suitably fresh b we have $(\mathsf{b}\ \ \mathsf{a}) \cdot e = (\mathsf{b}\ \ \mathsf{a}) \cdot e'$ and therefore $e = e'$.

□

There is a name abstraction operation for elements of families

$$\frac{e \in \mathbf{Nom}(X * \mathbb{A}_N \vdash E)}{\mathrm{abs}\, e \in \mathbf{Nom}(X \vdash И_N E)} \tag{40}$$

which is well-defined by

$$\mathrm{abs}\, e\, x \triangleq \langle \mathsf{a} \rangle_x e(x, \mathsf{a}) \text{ where } \mathsf{a} \# x \tag{41}$$

since the right-hand side is independent of the choice of $\mathsf{a} \in \mathbb{A}_N$ satisfying $\mathsf{a} \# x$ and does give a dependently equivariant function.

We next lift the process of concreting a name abstraction at a fresh atom to an operation in the CwF **Nom**. This involves using the version of name restriction given by Proposition 3.5. Suppose $f \in \mathbf{Nom}(X \vdash \mathit{И}_N E)$ and $a \in \mathbf{Nom}(X \vdash \mathbb{A}_N)$ satisfy $a \#_X f$ and hence also $a \#_X \mathit{И}_N E$. From the latter it follows that $E \in \mathbf{Nom}(X * \mathbb{A}_N)$ satisfies $a[\mathrm{p}] \#_{X*\mathbb{A}_N} E$ and hence by Proposition 3.4 that $\nu \#_{X*\mathbb{A}_N} (a[\mathrm{p}]\ \ \nu) \cdot E$. So as in Proposition 3.5 we can form $\mathrm{res}\,(a[\mathrm{p}]\ \ \nu) \cdot E \in \mathbf{Nom}(X)$. For each $x \in X$, picking any $\mathsf{b} \# x$, by Lemma 3.8 we get $(f\,x) @_x \mathsf{b} \in E_{(x,\mathsf{b})}$ and hence $(a\,x\ \ \mathsf{b}) \cdot ((f\,x) @_x \mathsf{b}) \in E_{((a\,x\ \ \mathsf{b})\cdot x, a\,x)} = (\mathrm{res}\,(a[\mathrm{p}]\ \ \nu) \cdot E)_x$, by (37). Furthermore, this element of $(\mathrm{res}\,(a[\mathrm{p}]\ \ \nu) \cdot E)_x$ is independent of the choice of of b, because $a \#_X f$. To see this, suppose $\mathsf{b}' \# x$ and let $e \triangleq (f\,x) @_x \mathsf{b}$ and $e' \triangleq (f\,x) @_x \mathsf{b}'$. Thus $f\,x = \langle \mathsf{b} \rangle_x e = \langle \mathsf{b}' \rangle_x e'$. From $a \#_X f$, that is $(a[\mathrm{p}]\ \ \nu) \cdot f[\mathrm{p}] = f[\mathrm{p}]$, we get $f((a\,x\ \ \mathsf{b}) \cdot x) = f\,x = f((a\,x\ \ \mathsf{b}') \cdot x)$ and hence $\langle a\,x \rangle_x (a\,x\ \ \mathsf{b}) \cdot e = (a\,x\ \ \mathsf{b}) \cdot f\,x = \langle a\,x \rangle_x (a\,x\ \ \mathsf{b}') \cdot e'$; therefore $(a\,x\ \ \mathsf{b}) \cdot e = (a\,x\ \ \mathsf{b}') \cdot e'$, by definition of $\approx_x$.

In this way we get an operation

$$\frac{f \in \mathbf{Nom}(X \vdash \mathit{И}_N E) \qquad a \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad a \#_X f}{\mathrm{conc}\, f\, a \in \mathbf{Nom}(X \vdash \mathrm{res}\,(a[\mathrm{p}]\ \ \nu) \cdot E)} \tag{42}$$

well-defined by:

$$\mathrm{conc}\, f\, a\, x \triangleq (a\,x\ \ \mathsf{b}) \cdot (f x @_x \mathsf{b}) \text{ where } \mathsf{b} \# x \tag{43}$$

It is not hard to see that $\mathit{И}_N$, abs and conc are stable under re-indexing

$$(\mathit{И}_N E)[g] = \mathit{И}_N (E[g * \mathbb{A}_N]) \tag{44}$$
$$(\mathrm{abs}\, e)[g] = \mathrm{abs}(e[g * \mathbb{A}_N]) \tag{45}$$
$$(\mathrm{conc}\, f\, a)[g] = \mathrm{conc}(f[g])(a[g]) \tag{46}$$

and satisfy the following forms of β- and η-conversion:

$$\mathrm{conc}(\mathrm{abs}\, e)\, a = \mathrm{res}((a[\mathrm{p}]\ \ \nu) \cdot e) \tag{47}$$
$$\mathrm{abs}(\mathrm{conc}(f[\mathrm{p}])\, \nu) = f \tag{48}$$

Remark 3.9 (adjoint characterization of $\mathit{И}_N$) The non-dependent name abstraction $[\mathbb{A}_N]Y$ discussed in [16, Sect. 4.7] is the special case of (39) when $X = 1$ and $E = Y[\mathrm{p}]$ for some $Y \in \mathbf{Nom}(1) \cong \mathbf{Nom}$. The functor $[\mathbb{A}_N](\_) : \mathbf{Nom} \to \mathbf{Nom}$ can be characterised as the right adjoint to the separated product function $(\_) * \mathbb{A}_N : \mathbf{Nom} \to \mathbf{Nom}$; see [16, Theorem 4.12]. Similarly, $\mathit{И}_N E$ can be given a characterization as a right adjoint, as follows:

For each $X \in \mathbf{Nom}$ there is a functor $(\_) * \mathbb{A}_N : \mathbf{Nom}(X) \to \mathbf{Nom}(X * \mathbb{A}_N)$ which takes an object $E \in \mathbf{Nom}(X)$ to the family given by

$$(E * \mathbb{A}_N)_{(x,\mathsf{a})} \triangleq \{e \in E_x \mid \mathsf{a} \# (x, e) \in X.E\} \qquad ((x, \mathsf{a}) \in X * \mathbb{A}_N) \tag{49}$$

with dependent G-action inherited from that for E: $\pi \in G, e \in (E*\mathbb{A}_N)_{(x,\mathsf{a})} \mapsto \pi \cdot e \in (E*\mathbb{A}_N)_{(\pi\cdot x, \pi\,\mathsf{a})}$. The functor takes morphisms $f \in \mathbf{Nom}(X)(E, E') = \mathbf{Nom}(X.E \vdash$

$E'[\mathrm{p}])$ to $f * \mathbb{A}_N \in \mathbf{Nom}(X * \mathbb{A}_N)(E *_N \mathbb{A}_N, E' * \mathbb{A}_N) = \mathbf{Nom}((X * \mathbb{A}_N).(E * \mathbb{A}_N) \vdash (E' * \mathbb{A}_N)[\mathrm{p}])$, where

$$(f * \mathbb{A}_N)((x, \mathsf{a}), e) \triangleq f(x, e) \qquad (((x, \mathsf{a}), e) \in (X * \mathbb{A}_N).(E * \mathbb{A}_N)) \tag{50}$$

(which is well-defined, because f is dependently equivariant and hence satisfies $\mathsf{a} \# (x, e) \in X.E \Rightarrow \mathsf{a} \# (x, f(x, e)) \in X.E'$). The functor $(\_) * \mathbb{A}_N : \mathbf{Nom}(X) \to \mathbf{Nom}(X * \mathbb{A}_N)$ is in fact full and faithful: faithfulness is immediate (since for any $(x, e) \in X.E$ we can always find some $\mathsf{a} \in \mathbb{A}_N$ with $\mathsf{a} \# (x, e)$); fullness amounts to the fact that since $g \in \mathbf{Nom}((X * \mathbb{A}_N).(E * \mathbb{A}_N) \vdash (E' * \mathbb{A}_N)[\mathrm{p}])$ satisfies

$$\mathsf{a} \# (x, e) \in X.E \;\Rightarrow\; \mathsf{a} \# (x, g((x, \mathsf{a}), e)) \in X.E'$$

we get a well-defined $f \in \mathbf{Nom}(X.E \vdash E'[\mathrm{p}])$ by defining

$$f(x, e) \triangleq g((x, \mathsf{a}), e) \text{ where } \mathsf{a} \# (x, e)$$

and $f * \mathbb{A}_N = g$.

We claim that the dependent name abstraction $И_N E$ is the value at $E \in \mathbf{Nom}(X)$ of a right adjoint to $(\_) * \mathbb{A}_N : \mathbf{Nom}(X) \to \mathbf{Nom}(X * \mathbb{A}_N)$. The counit of the adjunction at E

$$\varepsilon_E \in \mathbf{Nom}(X * \mathbb{A}_N)(И_N E * \mathbb{A}_N, E) = \mathbf{Nom}((X * \mathbb{A}_N).(И_N E * \mathbb{A}_N) \vdash E[\mathrm{p}])$$

is given by the concretion operation from Lemma 3.8(ii):

$$\varepsilon_E((x, \mathsf{a}), f) \triangleq f \mathbin{@_x} \mathsf{a} \qquad (((x, \mathsf{a}), f) \in (X * \mathbb{A}_N).(И_N E * \mathbb{A}_N))$$

This is an isomorphism (necessarily, because $(\_) * \mathbb{A}_N$ is full and faithful), whose inverse is given by the element of $\mathbf{Nom}((X * \mathbb{A}_N).E \vdash (И_N E)[\mathrm{p}])$ that maps each $((x, \mathsf{a}), e) \in (X * \mathbb{A}_N).E$ to $\langle \mathsf{a} \rangle_x e \in (И_N E)_x$. It has the universal property needed for the right adjoint at E: for each

$$f \in \mathbf{Nom}(X * \mathbb{A}_N)(E' * \mathbb{A}_N, E) = \mathbf{Nom}((X * \mathbb{A}_N).(E' * \mathbb{A}_N) \vdash E[\mathrm{p}])$$

we have

$$\hat{f} \in \mathbf{Nom}(X)(E', И_N E) = \mathbf{Nom}(X.E' \vdash (И_N E)[\mathrm{p}])$$

well-defined by

$$\hat{f}(x, e') \triangleq \langle \mathsf{a} \rangle_x f((x, \mathsf{a}), e') \text{ where } \mathsf{a} \# (x, e') \in X.E'$$

Then $\varepsilon_E \circ (\hat{f} * \mathbb{A}_N) = f$, because concretion satisfies $(\langle \mathsf{a} \rangle_x e) @_x \mathsf{a} = e$ (by Lemma 3.8); and $\hat{f}$ is the unique such morphism in $\mathbf{Nom}(X)(E', И_N E)$, because concretion satisfies $f = \langle \mathsf{a} \rangle_x (f \mathbin{@_x} \mathsf{a})$ (by the same lemma).

$$\frac{}{[\![\diamond]\!] \searrow 1 \in \mathbf{C}} \qquad \frac{[\![\Gamma]\!] \searrow X \in \mathbf{C} \qquad [\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \qquad x \notin \mathrm{dom}(\Gamma)}{[\![\Gamma(x:T)]\!] \searrow X.E \in \mathbf{C}}$$

$$\frac{[\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \qquad [\![\Gamma(x:T) \vdash T']\!] \searrow E' \in \mathbf{C}(X.E)}{[\![\Gamma \vdash \Pi(x:T)T']\!] \searrow \Pi\, E\, E' \in \mathbf{C}(X)}$$

$$\frac{[\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \qquad x \notin \mathrm{dom}(\Gamma)}{[\![\Gamma(x:T) \vdash x]\!] \searrow \mathrm{v} \in \mathbf{C}(X.E \vdash E[\mathrm{p}])}$$

$$\frac{[\![\Gamma \vdash x]\!] \searrow e \in \mathbf{C}(X \vdash E) \qquad x' \notin \mathrm{dom}(\Gamma) \qquad [\![\Gamma \vdash T']\!] \searrow E' \in \mathbf{C}(X)}{[\![\Gamma(x':T') \vdash x]\!] \searrow e[\mathrm{p}] \in \mathbf{C}(X.E' \vdash E[\mathrm{p}])}$$

$$\frac{[\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \qquad [\![\Gamma(x:T) \vdash t]\!] \searrow e \in \mathbf{C}(X.E \vdash F)}{[\![\Gamma \vdash \lambda(x:T)t]\!] \searrow \mathrm{lam}\, e \in \mathbf{C}(X \vdash \Pi\, E\, F)}$$

$$\frac{[\![\Gamma \vdash t]\!] \searrow f \in \mathbf{C}(X \vdash \Pi\, E\, F) \qquad [\![\Gamma \vdash t']\!] \searrow e \in \mathbf{C}(X \vdash E)}{[\![\Gamma \vdash t\, t']\!] \searrow \mathrm{app}\, f\ e \in \mathbf{C}(X \vdash F[\langle \mathrm{id}\, , e \rangle])}$$

Fig. 5. Semantics of MLTT in a CwF

4 Nominal Set Semantics of FreshMLTT

Hofmann [11, Sect. 3.5] (following Streicher [22]) gives the semantics of Martin-Löf Type Theory in a CwF $\mathbf{C}$. It takes the form of a partial interpretation function $[\![\_]\!]$ mapping contexts to objects, types-in-context to families and terms-in-context to elements of families. For simplicity we just consider Π-types, but other constructs of conventional Martin-Löf Type Theory can be interpreted similarly. Specifically, given a CwF $\mathbf{C}$ with Π-types (see Fig. 4), Fig. 5 inductively defines the graph of the interpretation function using relations of the following form, where Γ ranges over context expressions, T over type expressions, t over term expressions, X over $\mathbf{C}$-objects, E over $\mathbf{C}$-families and e over elements of $\mathbf{C}$-families:

$$\begin{array}{c} [\![\Gamma]\!] \searrow X \in \mathbf{C} \\ [\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \\ [\![\Gamma \vdash t]\!] \searrow e \in \mathbf{C}(X \vdash E) \end{array}$$

Here, we take $[\![\Gamma]\!] \searrow X \in \mathbf{C}$ to mean that $[\![\Gamma]\!]$ is defined (written as $[\![\Gamma]\!]{\downarrow}$) and equal to $X \in \mathbf{C}$, etc. Hofmann [11, Theorem 3.35] sketches the proof of the following soundness properties of the interpretation with respect to the provable judgements

of Martin-Löf Type Theory:

$$\Gamma \vdash \mathrm{ok} \;\Rightarrow\; (\exists X)\, [\![\Gamma]\!] \searrow X \in \mathbf{C} \tag{51}$$

$$\Gamma \vdash T \;\Rightarrow\; (\exists X, E)\, [\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \tag{52}$$

$$\Gamma \vdash t : T \;\Rightarrow\; (\exists X, E, e)\, [\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \;\wedge\; [\![\Gamma \vdash t]\!] \searrow e \in \mathbf{C}(X \vdash E) \tag{53}$$

$$\Gamma \vdash T = T' \;\Rightarrow\; (\exists X, E)\, [\![\Gamma \vdash T]\!] \searrow E \in \mathbf{C}(X) \;\wedge\; [\![\Gamma \vdash T']\!] \searrow E \in \mathbf{C}(X) \tag{54}$$

$$\Gamma \vdash t = t' : T \;\Rightarrow\; (\exists X, E, e)\, [\![\Gamma \vdash t]\!] \searrow e \in \mathbf{C}(X \vdash E) \;\wedge\; [\![\Gamma \vdash t']\!] \searrow e \in \mathbf{C}(X \vdash E) \tag{55}$$

Taking $\mathbf{C}$ to be the CwF **Nom** discussed in Sect. 3, the interpretation partial function can be extended to the expressions of FreshMLTT. Fig. 6 gives the additional rules. Types of names are interpreted as name objects (Sect. 3.3) and dependent name abstraction types by the operation (39). Name swapping on types and terms is interpreted using the operations in Sect. 3.4. Locally scoping names in types and terms is interpreted using (33) and (34). Terms for branching on name equality, for name abstraction and concretion are interpreted using (21), (40) and (42).

Lemma 4.1 (weakening) *If* $[\![\Gamma\Gamma']\!]\downarrow \in \mathbf{Nom}$ *and* $[\![\Gamma\Delta\Gamma']\!]\downarrow \in \mathbf{Nom}$, *we can define a generalised projection morphism* $\mathrm{P}(\Gamma, \Delta, \Gamma') \in \mathbf{Nom}([\![\Gamma\Delta\Gamma']\!], [\![\Gamma\Gamma']\!])$ *as follows:*

$$\begin{aligned}
\mathrm{P}(\Gamma, \diamond, \diamond) &= \mathrm{id} \\
\mathrm{P}(\Gamma, \Delta(x : T), \diamond) &= \mathrm{P}(\Gamma, \Delta, \diamond) \circ \mathrm{p} \\
\mathrm{P}(\Gamma, \Delta[n : N], \diamond) &= \mathrm{P}(\Gamma, \Delta, \diamond) \circ \mathrm{p} \\
\mathrm{P}(\Gamma, \Delta, \Gamma'(x : T)) &= \langle \mathrm{P}(\Gamma, \Delta, \Gamma') \circ \mathrm{p}\,, \mathrm{v} \rangle \\
\mathrm{P}(\Gamma, \Delta, \Gamma'[n : N]) &= \mathrm{P}(\Gamma, \Delta, \Gamma') * \mathbb{A}_N
\end{aligned}$$

Then for any term t *and any type* T, *we have*[10]

$$\begin{aligned}
[\![\Gamma\Delta\Gamma' \vdash T]\!] &\equiv [\![\Gamma\Gamma' \vdash T]\!][\mathrm{P}(\Gamma, \Delta, \Gamma')] \\
[\![\Gamma\Delta\Gamma' \vdash t]\!] &\equiv [\![\Gamma\Gamma' \vdash t]\!][\mathrm{P}(\Gamma, \Delta, \Gamma')]
\end{aligned}$$

Furthermore, the generalised projection morphisms $\mathrm{P}(\Gamma, \diamond[n : N], \Gamma')$ *are surjective, since* $\mathrm{p} \in \mathbf{Nom}(X * \mathbb{A}_N, X)$ (26) *is surjective (because given* $x \in X$ *we can always pick some atom* a *not in its support and hence with* $\mathrm{p}(x, a) = x$*).* □

Theorem 4.2 (soundness) *The soundness properties* (51)–(55) *of the interpretation of Martin-Löf Type Theory in* **Nom** *continue to hold when it is extended to FreshMLTT. In addition, the following soundness properties for definitional fresh-*

[10] We write $x \equiv y$ the Kleene equality of two potentially undefined expressions, i.e. if one side is defined so is the other and then $x = y$.

$$\frac{[\![\Gamma]\!] \searrow X \in \mathbf{Nom} \qquad n \notin \mathrm{dom}(\Gamma)}{[\![\Gamma[n:N]]\!] \searrow X * \mathbb{A}_N \in \mathbf{Nom}} \qquad \frac{[\![\Gamma]\!] \searrow X \in \mathbf{Nom}}{[\![\Gamma \vdash N]\!] \searrow \mathbb{A}_N \in \mathbf{Nom}(X)}$$

$$\frac{[\![\Gamma \vdash x]\!] \searrow e \in \mathbf{Nom}(X \vdash E) \qquad n' \notin \mathrm{dom}(\Gamma)}{[\![\Gamma[n':N'] \vdash x]\!] \searrow e[\mathrm{p}] \in \mathbf{Nom}(X * \mathbb{A}_{N'} \vdash E[\mathrm{p}])}$$

$$\frac{[\![\Gamma]\!] \searrow X \in \mathbf{Nom} \qquad n \notin \mathrm{dom}(\Gamma)}{[\![\Gamma[n:N] \vdash n]\!] \searrow \nu \in \mathbf{Nom}(X * \mathbb{A}_N \vdash \mathbb{A}_N)}$$

$$\frac{[\![\Gamma \vdash n]\!] \searrow e \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad x' \notin \mathrm{dom}(\Gamma) \qquad [\![\Gamma \vdash T']\!] \searrow E' \in \mathbf{Nom}(X)}{[\![\Gamma(x':T') \vdash n]\!] \searrow e[\mathrm{p}] \in \mathbf{Nom}(X.E' \vdash \mathbb{A}_N)}$$

$$\frac{[\![\Gamma \vdash n]\!] \searrow e \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad n' \notin \mathrm{dom}(\Gamma)}{[\![\Gamma[n':N'] \vdash n]\!] \searrow e[\mathrm{p}] \in \mathbf{Nom}(X * \mathbb{A}_{N'} \vdash \mathbb{A}_N)}$$

$$\frac{\begin{array}{c}[\![\Gamma \vdash t]\!] \searrow a \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad [\![\Gamma \vdash n]\!] \searrow b \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \\ [\![\Gamma \vdash t_1]\!] \searrow e \in \mathbf{Nom}(X \vdash E) \qquad [\![\Gamma \vdash t_2]\!] \searrow f \in \mathbf{Nom}(X \vdash E)\end{array}}{[\![\Gamma \vdash \mathsf{if}\ t = n\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2]\!] \searrow \mathrm{ifeq}(a,b,e,f) \in \mathbf{Nom}(X \vdash E)}$$

$$\frac{\begin{array}{c}[\![\Gamma \vdash T]\!] \searrow E \in \mathbf{Nom}(X) \\ [\![\Gamma \vdash n]\!] \searrow a \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad [\![\Gamma \vdash n']\!] \searrow b \in \mathbf{Nom}(X \vdash \mathbb{A}_N)\end{array}}{[\![\Gamma \vdash (n\ \ n') \cdot T]\!] \searrow (a\ \ b) \cdot E \in \mathbf{Nom}(X)}$$

$$\frac{\begin{array}{c}[\![\Gamma \vdash t]\!] \searrow e \in \mathbf{Nom}(X \vdash E) \\ [\![\Gamma \vdash n]\!] \searrow a \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad [\![\Gamma \vdash n']\!] \searrow b \in \mathbf{Nom}(X \vdash \mathbb{A}_N)\end{array}}{[\![\Gamma \vdash (n\ \ n') \cdot t]\!] \searrow (a\ \ b) \cdot e \in \mathbf{Nom}(X \vdash (a\ \ b) \cdot E)}$$

$$\frac{[\![\Gamma[n:N] \vdash T]\!] \searrow E \in \mathbf{Nom}(X * \mathbb{A}_N) \qquad \nu \#_{X*\mathbb{A}_N} E}{[\![\Gamma \vdash \nu[n:N]\,T]\!] \searrow \mathrm{res}\,E \in \mathbf{Nom}(X)}$$

$$\frac{[\![\Gamma[n:N] \vdash t]\!] \searrow e \in \mathbf{Nom}(X * \mathbb{A}_N \vdash E) \qquad \nu \#_{X*\mathbb{A}_N} e}{[\![\Gamma \vdash \nu[n:N]\,t]\!] \searrow \mathrm{res}\,e \in \mathbf{Nom}(X \vdash \mathrm{res}\,E)}$$

$$\frac{[\![\Gamma[n:N] \vdash T]\!] \searrow E \in \mathbf{Nom}(X * \mathbb{A}_N)}{[\![\Gamma \vdash \mathit{И}[n:N]T]\!] \searrow \mathit{И}_N E \in \mathbf{Nom}(X)}$$

$$\frac{[\![\Gamma[n:N] \vdash t]\!] \searrow e \in \mathbf{Nom}(X * \mathbb{A}_N \vdash E)}{[\![\Gamma \vdash \alpha[n:N]t]\!] \searrow \mathrm{abs}\,e \in \mathbf{Nom}(X \vdash \mathit{И}_N E)}$$

$$\frac{[\![\Gamma \vdash t]\!] \searrow f \in \mathbf{Nom}(X \vdash \mathit{И}_N E) \qquad [\![\Gamma \vdash n]\!] \searrow a \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \qquad a \#_X f}{[\![\Gamma \vdash t\ @\ n]\!] \searrow \mathrm{conc}\,f\,a \in \mathbf{Nom}(X \vdash \mathrm{res}\,(a[\mathrm{p}]\ \ \nu) \cdot E)}$$

Fig. 6. Additional rules for the semantics of FreshMLTT

ness hold with respect to the relations $a \#_X E$ and $a \#_X e$ from Definition 3.3:

$$\Gamma \vdash (n : N) \# T \;\Rightarrow\; (\exists X, a, E)\, [\![\Gamma \vdash n]\!] \searrow a \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \wedge [\![\Gamma \vdash T]\!] \searrow E \in \mathbf{Nom}(X) \;\wedge\; a \#_X E \quad (56)$$

$$\Gamma \vdash (n : N) \# t : T \;\Rightarrow\; (\exists X, a, E, e)\, [\![\Gamma \vdash n]\!] \searrow a \in \mathbf{Nom}(X \vdash \mathbb{A}_N) \wedge [\![\Gamma \vdash T]\!] \searrow E \in \mathbf{Nom}(X) \;\wedge\; [\![\Gamma \vdash t]\!] \searrow e \in \mathbf{Nom}(X \vdash E) \;\wedge\; a \#_X e \quad (57)$$

Proof. (sketch) The proof is done by mutual induction on the judgements of the FreshMLTT type system. We omit the induction steps for those judgements that already appear in MLTT, since a detailed proof (using contextual categories) can be found in [22, Chapter 3]. The remaining induction steps follow from the properties of the CwF **Nom** established in Sect. 3. We give details for the interesting cases.

Case

$$\frac{\Gamma\Delta \vdash T \qquad \Gamma\Delta \vdash T' \qquad \Gamma[n : N]\Delta \vdash T = T'}{\Gamma\Delta \vdash T = T'}\ (\textsc{n-elim-typ}) \qquad \frac{\Gamma\Delta \vdash t : T \qquad \Gamma\Delta \vdash t' : T \qquad \Gamma[n : N]\Delta \vdash t = t' : T}{\Gamma\Delta \vdash t = t' : T}\ (\textsc{n-elim-term})$$

Assuming $\Gamma\Delta \vdash T$ and $\Gamma\Delta \vdash T'$, we apply the induction hypothesis to obtain $[\![\Gamma\Delta]\!] \searrow X \in \mathbf{Nom}$, $[\![\Gamma\Delta \vdash T]\!] \searrow E \in \mathbf{Nom}(X)$ and $[\![\Gamma\Delta \vdash T']\!] \searrow E' \in \mathbf{Nom}(X)$. Assuming also $\Gamma[n : N]\Delta \vdash T = T'$, gives us $[\![\Gamma[n : N]\Delta]\!] \searrow X' \in \mathbf{Nom}$ and $E[\mathrm{P}(\Gamma, \diamond[n : N], \Delta)] = E'[\mathrm{P}(\Gamma, \diamond[n : N], \Delta)] \in \mathbf{Nom}(X')$ using the Weakening Lemma (4.1). Then $E = E' \in \mathbf{Nom}(X)$ follows from the fact that projection morphisms are surjective.

The induction step for (n-elim-term) similarly follows from the surjectivity of projection morphisms.

Case

$$\frac{\Gamma \vdash n : N \qquad \Gamma \vdash T \qquad \Gamma[n' : N] \vdash (n\ \ n') \cdot T = T}{\Gamma \vdash (n : N) \# T}\ (\textsc{fresh-typ})$$

Supposing $\Gamma \vdash n : N$ and $\Gamma \vdash T$, we obtain $[\![\Gamma]\!] \searrow X \in \mathbf{Nom}$, $[\![\Gamma \vdash n]\!] \searrow a \in \mathbf{Nom}(X \vdash \mathbb{A}_N)$ and $[\![\Gamma \vdash T]\!] \searrow E \in \mathbf{Nom}(X)$ using the induction hypothesis.

It follows from the Weakening Lemma that $[\![\Gamma[n' : N] \vdash T]\!] \searrow E[\mathrm{p}] \in \mathbf{Nom}(X * \mathbb{A}_n)$. Therefore

$$[\![\Gamma[n' : N] \vdash (n\ \ n') \cdot T]\!] \searrow (a[\mathrm{p}]\ \ \nu) \cdot E[\mathrm{p}] \in \mathbf{Nom}(X * \mathbb{A}_N)$$

with $(a[\mathrm{p}]\ \ \nu) \cdot E[\mathrm{p}] = E[\mathrm{p}]$, i.e. $a \#_X E$.

Case

$$\frac{\Gamma \vdash T \qquad \Gamma\Delta \vdash \mathrm{ok} \qquad [n : N], [n' : N] \in \Delta}{\Gamma\Delta \vdash (n\ \ n') \cdot T = T}\ (\textsc{fresh-hyp-typ})$$

Suppose that $\Gamma \vdash T$, $\Gamma\Delta \vdash \mathrm{ok}$ and $[n : N], [n' : N] \in \Delta$. We only consider the case where $n \neq n'$ and $\Delta = \Delta_1[n : N]\Delta_2[n' : N]\Delta_3$. Using the induction hypothesis we get $[\![\Gamma]\!]{\downarrow}, [\![\Delta]\!]{\downarrow}, [\![\Gamma\Delta]\!]{\downarrow} \in \mathbf{Nom}$ and $[\![\Gamma \vdash T]\!] \searrow E \in \mathbf{Nom}([\![\Gamma]\!])$. We have $\nu \#_{[\![\Gamma\Delta_1]\!]*\mathbb{A}_N} E[\mathrm{P}(\Gamma, \Delta_1, \diamond)][p]$ by (32) and hence $a \#_{\Gamma\Delta} E'$ via the Weakening Lemma, where $a \triangleq \nu[\mathrm{P}(\Gamma\Delta_1[n : N], \Delta_2[n' : N]\Delta_3, \diamond)]$ and $E' \triangleq E[\mathrm{P}(\Gamma, \Delta, \diamond)]$. Similarly, $a' \#_{\Gamma\Delta} E'$ with $a' \triangleq \nu[\mathrm{P}(\Gamma\Delta_1[n : N]\Delta_2[n' : N], \Delta_3, \diamond)]$. It follows from Definition 3.3 and the properties of dependent G-actions that

$$\begin{aligned} E'[p] &= (a'[\mathrm{p}]\ \ \nu) \cdot (a[\mathrm{p}]\ \ \nu) \cdot E'[\mathrm{p}] \\ &= (a[\mathrm{p}]\ \ a'[\mathrm{p}]) \cdot (a'[\mathrm{p}]\ \ \nu) \cdot E'[\mathrm{p}] \\ &= (a[\mathrm{p}]\ \ a'[\mathrm{p}]) \cdot E[\mathrm{p}] \\ &= ((a\ \ a') \cdot E')[\mathrm{p}] \end{aligned}$$

Since the projection function, p, is surjective, $(a\ \ a') \cdot E'$ and E' are equal families over $[\![\Gamma\Delta]\!] \in \mathbf{Nom}$.

Case

$$\frac{\Gamma[n : N] \vdash (n : N)\,\#\,T}{\Gamma[n : N] \vdash \nu[n : N]\,T = T}\ (\textsc{local-typ-comp})$$

Suppose that $\Gamma[n : N] \vdash (n : N)\,\#\,T$. Then we obtain $[\![\Gamma[n : N]]\!] \searrow X * \mathbb{A}_N \in \mathbf{Nom}$ and $[\![\Gamma[n : N] \vdash T]\!] \searrow E \in \mathbf{Nom}(X * \mathbb{A}_N)$ satisfying $\nu \#_{X*\mathbb{A}_N} E$ by applying the induction hypothesis and hence $[\![\Gamma \vdash \nu[n : N]T]\!] \searrow \mathrm{res}\, E \in \mathbf{Nom}(X)$. Consequently, $[\![\Gamma[n : N] \vdash \nu[n : N]T]\!] \searrow (\mathrm{res}\, E)[p] \in \mathbf{Nom}(X * \mathbb{A}_N)$ by the Weakening Lemma and $(\mathrm{res}\, E)[p] = E \in \mathbf{Nom}(X * \mathbb{A}_N)$ by (35).

Case

$$\frac{\Gamma[n' : N] \vdash (n : N)\,\#\,T \qquad \Gamma \vdash (n : N)\,\#\,t : \text{И}[n' : N]T}{\Gamma \vdash t\,@\,n : \nu[n' : N]\,(n\ \ n') \cdot T}\ (\text{И}\textsc{-elim})$$

Suppose $\Gamma[n' : N] \vdash (n : N)\,\#\,T$. Applying the induction hypothesis gives $[\![\Gamma[n' : N]]\!] \searrow X * \mathbb{A}_N \in \mathbf{Nom}$, $[\![\Gamma[n' : N] \vdash T]\!] \searrow E \in \mathbf{Nom}(X * \mathbb{A}_N)$, $[\![\Gamma[n' : N] \vdash n]\!] \searrow a[p] \in \mathbf{Nom}(X * \mathbb{A}_N \vdash \mathbb{A}_N)$ and $a[p] \#_{X*\mathbb{A}_N} E$. It follows from (32) that $\nu \#_{x*\mathbb{A}_N} (a[p]\ \ \nu) \cdot E$. Hence, $[\![\Gamma \vdash \nu[n : n'](n\ \ n') \cdot T]\!] \searrow \mathrm{res}\,(a[p]\ \ \nu) \cdot E \in \mathbf{Nom}(X)$ by (33).

Supposing further that $\Gamma \vdash (n : N) \# t : \mathsf{И}[n' : N]T$, we obtain $[\![\Gamma \vdash \mathsf{И}[n' : N]T]\!] \searrow \mathsf{И}_N[\![\Gamma[n' : N]T]\!] = \mathsf{И}_N E \in \mathbf{Nom}(X)$ and $[\![\Gamma \vdash t]\!] \searrow f \in \mathbf{Nom}(X \vdash \mathsf{И}_N E)$ with $a \#_X f$ from the induction hypothesis. Hence, $[\![\Gamma \vdash t \mathbin{@} n]\!] \searrow \mathrm{conc}\, f a \in \mathbf{Nom}(X \vdash \mathrm{res}\,(a[p]\ \ \nu) \cdot E)$.

Case

$$\frac{\Gamma[n' : N] \vdash (n : N) \# t : T \qquad n \neq n'}{\Gamma \vdash (\alpha[n' : N]t) \mathbin{@} n = \nu[n' : N]\,(n\ \ n') \cdot t : \nu[n' : N]\,(n\ \ n') \cdot T}\ (\mathsf{И}\text{-comp})$$

Suppose $\Gamma[n' : N] \vdash (n : N) \# t : T$ with $n \neq n'$. The induction hypothesis provides us with $[\![\Gamma[n' : N]]\!] \searrow X * \mathbb{A}_N$, $[\![\Gamma[n' : N] \vdash n]\!] \searrow a[p] \in \mathbf{Nom}(X * \mathbb{A}_N \vdash \mathbb{A}_N)$, $[\![\Gamma[n' : N] \vdash T]\!] \searrow E \in \mathbf{Nom}(X * \mathbb{A}_N)$ and $[\![\Gamma[n' : N] \vdash t]\!] \searrow e \in \mathbf{Nom}(X * \mathbb{A}_N \vdash E)$ satisfying $a[p] \#_{X * \mathbb{A}_N} e$. It follows from (40) and (42) that $\mathrm{conc}(\mathrm{abs}\, e)\, a \in \mathbf{Nom}(X \vdash \mathrm{res}\,(a[p]\ \ \nu) \cdot E)$ and from (31) and (34) that $\mathrm{res}\,(a[p]\ \ \nu) \cdot e \in \mathbf{Nom}(X \vdash \mathrm{res}\,(a[p]\ \ \nu) \cdot E)$. Hence, $\mathrm{conc}(\mathrm{abs}\, e)\, a = \mathrm{res}\,(a[p]\ \ \nu) \cdot e$ by (47).

Case

$$\frac{\Gamma \vdash t : \mathsf{И}[n : N]T \qquad n \notin \mathrm{fn}(t)}{\Gamma \vdash t = \alpha[n : N](t \mathbin{@} n) : \mathsf{И}[n : N]T}\ (\mathsf{И}\text{-uniq})$$

Suppose $\Gamma \vdash t : \mathsf{И}[n : N]T$. We obtain $[\![\Gamma]\!] \searrow X \in \mathbf{Nom}$, $[\![\Gamma \vdash \mathsf{И}[n : N]T]\!] \searrow \mathsf{И}_N E \in \mathbf{Nom}(X)$ and $[\![\Gamma \vdash t]\!] \searrow f \in \mathbf{Nom}(X \vdash \mathsf{И}_N E)$ by applying the induction hypothesis. Then $[\![\Gamma[n : N] \vdash t]\!] \searrow f[p] \in \mathbf{Nom}(X * \mathbb{A}_N \vdash \mathsf{И}_N(E[p * \mathbb{A}_N]))$ by the Weakening Lemma and $\mathrm{conc}(f[\mathrm{p}])\, \nu \in \mathbf{Nom}(X * \mathbb{A}_N \vdash \mathrm{res}\,(\nu[p]\ \ \nu) \cdot E[p * \mathbb{A}_N])$ by (42) using Proposition 3.4. We have $\mathrm{res}\,(\nu[p]\ \ \nu) \cdot E[p * \mathbb{A}_N] = E \in \mathbf{Nom}(X * \mathbb{A}_N)$ by (28) and hence $\mathrm{abs}(\mathrm{conc}(f[\mathrm{p}])\, \nu) \in \mathbf{Nom}(X \vdash \mathsf{И}_N E)$. Thus $\mathrm{abs}(\mathrm{conc}(f[\mathrm{p}])\, \nu) = f \in \mathbf{Nom}(X \vdash \mathsf{И}_N E)$ by (48). □

5 Related Work

The first work on a dependent type theory with features inspired by the nominal sets treatment of names and binding was by Schöpp and Stark [19,18]. They make use of a 'bunched' structure for typing contexts, whose semantics combines the usual cartesian product with the separated product $X * Y = \{(x, y) \in X \times Y \mid x \# y\}$ of nominal sets [16, Sect. 3.4]; and they develop an abstract view of separated products and its adjoints using fibred category theory, together with a corresponding version of Martin-Löf's extensional Type Theory [13]. Here we are less ambitious, since we only consider the special case of $X * Y$ when $Y = \mathbb{A}_N$ is a nominal set of names. We also use categories with families rather than fibred categories, but to a large extent that is a matter of personal taste (of the first author). However, we also target the intensional version of Martin-Löf Type Theory [14], aiming for a system that is not

only implementable, but also reasonably simple from a user point of view. This is also one of the aims of Cheney's DNTT [3], which was partly inspired by the work of Schöpp and Stark and is the system most closely related to the one presented here. Syntactically, DNTT adds to LF [10] names and dependent name abstraction types, with associated abstraction and concretion terms. However, the rule for forming concretions in DNTT has a strong hypothesis involving meta-theoretic freshness of the concreting name, rather than the 'definitional freshness' we have adopted from Crole and Nebel [7]. In our notation, the rule is

$$\frac{\Gamma \vdash t : \mathit{N}[n : N]T \qquad \Gamma[n : N]\Delta \vdash \mathrm{ok}}{\Gamma[n : N]\Delta \vdash t \mathbin{@} n : T}$$

and, for example, it is not strong enough to type the term j in Example 2.2. As explained in Sect. 2.3, the FreshMLTT rules for concretion involve extra syntactic constructs, namely terms for swapping names and for locally scoping names. The latter in particular should make FreshMLTT more expressive than existing 'nominal' type theories for encoding syntax-manipulating algorithms involving the creation of fresh names with a static scope – a common feature when it comes to manipulating binders with explicit names. Neither DNTT, nor the 'calculus of nominal inductive constructions' (CNIC) of Westbrook et al. [26,25] feature constructs for locally scoped names. [11] There are of course other approaches than the nominal one for dealing with issues to do with object-language binders in logical frameworks; we refer the reader to [3, Sect. 2] for a brief survey.

6 Future Work

We plan to develop a notion of normal form for FreshMLTT expressions, together with an algorithmic version of the rules and an implementation of them. In particular, we hope to show that FreshMLTT has decidable type-checking. So far we have normal forms and a (currently partial) normalization result via normalization by evaluation [1] for a simply typed version of FreshMLTT. For simplicity, we have presented our theory of dependently typed name abstraction within Martin-Löf's open-ended type theory. An implemented version will need a closed system; and to be reasonably expressive that system should have universes and inductively defined families of types. As the Agda system shows, the usefulness of the latter is much enhanced by introducing dependently typed patterns [6]. We do not yet know whether these can be enhanced with FreshML-style *name abstraction patterns* [20, Sect. 2.4]. In any case, the use of dependent name abstraction within inductively defined families of types seems to offer interesting possibilities. For example, the CwF **Nom** supports the interpretation of inductively defined types for *propositional freshness*, which relate to definitional freshness in the same way that propositional equality types relate to definitional equality. Such types should allow one to develop constructive versions of some theorems of nominal logic [15] via the Curry-Howard

[11] In particular, the ν types in CNIC are a form of name abstraction, not local scoping.

correspondence. As well inductive types, dually, it seems worth investigating the interaction between dependent name abstraction and coinductively defined types; see for example [12].

We have given a semantics for FreshMLTT by identifying appropriate structure in the particular CwF of nominal sets. It should be possible to axiomatize that structure as a generalised algebraic theory [2] extending the one for categories with families given by Dybjer [8, Sect. 2.2]. In this way one would obtain an algebraic version of the type theory we study in this paper.

References

[1] Berger, U. and H. Schwichtenberg, *An inverse of the evaluation functional for typed λ-calculus*, in: *6th Annual Symposium on Logic in Computer Science* (1991), pp. 203–211.

[2] Cartmell, J., *Generalised algebraic theories and contextual categories*, Annals of Pure and Applied Logic **32** (1986), pp. 209–243.

[3] Cheney, J., *A dependent nominal type theory*, Logical Methods in Computer Science **8** (2012), p. (1:08).

[4] Clouston, R. A., "Equational Logic for Names and Binders," Ph.D. thesis, University of Cambridge (2009).

[5] Clouston, R. A. and A. M. Pitts, *Nominal equational logic*, in: L. Cardelli, M. Fiore and G. Winskel, editors, *Computation, Meaning and Logic, Articles dedicated to Gordon Plotkin*, Electronic Notes in Theoretical Computer Science **172**, Elsevier, 2007 pp. 223–257.

[6] Coquand, T., *Pattern matching with dependent types*, in: B. Nordström, K. Petersson and G. D. Plotkin, editors, *Proceedings of the 1992 Workshop on Types for Proofs and Programs, Båstad, Sweden*, 1992, pp. 66–79.

[7] Crole, R. L. and F. Nebel, *An internal language for FM-cartesian closed categories (the nominal lambda calculus)* (2013), in Michael Mislove, editor, Proceedings of Mathematical Foundations of Programming Semantics (MFPS XXIX), New Orleans, LA, 2013. To appear.

[8] Dybjer, P., *Internal type theory*, in: S. Berardi and M. Coppo, editors, *Types for Proofs and Programs*, Lecture Notes in Computer Science **1158**, Springer Berlin Heidelberg, 1996 pp. 120–134.
URL http://dx.doi.org/10.1007/3-540-61780-9_66

[9] Gabbay, M. J. and A. Mathijssen, *Nominal (universal) algebra: Equational logic with names and binding*, Journal of Logic and Computation **19** (2009), pp. 1455–1508.

[10] Harper, R., F. Honsell and G. D. Plotkin, *A framework for defining logics*, Journal of the Association for Computing Machinery **40** (1993), pp. 143–184.

[11] Hofmann, M., *Syntax and semantics of dependent types*, in: A. M. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, Cambridge University Press, 1997 pp. 79–130.

[12] Kurz, A., D. L. Petrisan, P. Severi and F.-J. de Vries, *Nominal coalgebraic data types with applications to lambda calculus*, Logical Methods in Computer Science **9(4:20)** (2013).

[13] Martin-Löf, P., "Intuitionistic Type Theory," Bibliopolis, Napoli, 1984, 91 pp.

[14] Nordström, B., K. Petersson and J. M. Smith, "Programming in Martin-Löf's Type Theory," Oxford University Press, 1990.

[15] Pitts, A. M., *Nominal logic, a first order theory of names and binding*, Information and Computation **186** (2003), pp. 165–193.

[16] Pitts, A. M., "Nominal Sets: Names and Symmetry in Computer Science," Cambridge Tracts in Theoretical Computer Science **57**, Cambridge University Press, 2013.

[17] Pitts, A. M. and M. J. Gabbay, *A metalanguage for programming with bound names modulo renaming*, in: R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction. 5th International Conference, MPC2000, Ponte de Lima, Portugal, July 2000. Proceedings*, Lecture Notes in Computer Science **1837** (2000), pp. 230–255.

[18] Schöpp, U., "Names and Binding in Type Theory," Ph.D. thesis, University of Edinburgh (2006).

[19] Schöpp, U. and I. D. B. Stark, *A dependent type theory with names and binding*, in: *Computer Science Logic, CSL04, Karpacz, Poland*, Lecture Notes in Computer Science **3210** (2004), pp. 235–249.

[20] Shinwell, M. R. and A. M. Pitts, *Fresh Objective Caml user manual*, Technical Report UCAM-CL-TR-621, University of Cambridge Computer Laboratory (2005).

[21] Shinwell, M. R., A. M. Pitts and M. J. Gabbay, *FreshML: Programming with binders made simple*, in: *Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), Uppsala, Sweden* (2003), pp. 263–274.

[22] Streicher, T., "Semantics of Type Theory," Birkhäuser, 1991.

[23] Univalent Foundations Program, T., "Homotopy Type Theory: Univalent Foundations for Mathematics," http://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[24] Urban, C., A. M. Pitts and M. J. Gabbay, *Nominal unification*, Theoretical Computer Science **323** (2004), pp. 473–497.

[25] Westbrook, E., "Higher-Order Encodings with Constructors," Ph.D. thesis, Washington University in St Louis (2008).

[26] Westbrook, E., A. Stump and E. Austin, *The calculus of nominal inductive constructions: an intensional approach to encoding name-bindings*, in: *Proceedings of the Fourth International Workshop on Logical Frameworks and Meta-languages: Theory and Practice (LFMTP 2009), Montreal, Canada*, ACM International Conference Proceeding Series (2009), pp. 74–83.

A The Rules of FreshMLTT

We only give the rules involving the new features of FreshMLTT. For the usual rules of Martin-Löf Type Theory see [14], [11, Sect. 2], or [23, Appendix A.2].

Structural rules

$$\frac{\Gamma \vdash \mathrm{ok} \qquad \Gamma\Delta \vdash \mathcal{J} \qquad n \notin \mathrm{dom}(\Gamma\Delta)}{\Gamma[n:N]\Delta \vdash \mathcal{J}}\ (\textsc{wkg})$$

$$\frac{\Gamma\Delta \vdash T \qquad \Gamma\Delta \vdash T' \qquad \Gamma[n:N]\Delta \vdash T = T'}{\Gamma\Delta \vdash T = T'}\ (\textsc{n-elim-typ}) \qquad \frac{\Gamma\Delta \vdash t : T \qquad \Gamma\Delta \vdash t' : T \qquad \Gamma[n:N]\Delta \vdash t = t' : T}{\Gamma\Delta \vdash t = t' : T}\ (\textsc{n-elim-term})$$

Well-formed types

$$\frac{\Gamma \vdash \mathrm{ok}}{\Gamma \vdash N}\ (\textsc{n-form}) \qquad \frac{\Gamma \vdash n : N \qquad \Gamma \vdash n' : N \qquad \Gamma \vdash T}{\Gamma \vdash (n\ \ n') \cdot T}\ (\textsc{swap-typ})$$

$$\frac{\Gamma[n:N] \vdash (n:N) \# T}{\Gamma \vdash \nu[n:N]\,T}\ (\textsc{local-typ}) \qquad \frac{\Gamma[n:N] \vdash T}{\Gamma \vdash \text{И}[n:N]T}\ (\text{И}\textsc{-form})$$

Typing relation

$$\frac{\Gamma \vdash \mathrm{ok} \qquad [n:N] \in \Gamma}{\Gamma \vdash n : N}\ (\textsc{n-intro}) \qquad \frac{\Gamma \vdash n : N \qquad \Gamma \vdash n' : N \qquad \Gamma \vdash t : T}{\Gamma \vdash (n\ \ n') \cdot t : (n\ \ n') \cdot T}\ (\textsc{swap-term})$$

$$\frac{\Gamma \vdash t : N \qquad \Gamma \vdash n : N \qquad \Gamma \vdash t_1 : T \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash \mathsf{if}\ t = n\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : T}\ (\textsc{ifeq}) \qquad \frac{\Gamma[n:N] \vdash (n:N) \# t : T}{\Gamma \vdash \nu[n:N]\,t : \nu[n:N]\,T}\ (\textsc{local-term})$$

$$\frac{\Gamma[n:N] \vdash t : T}{\Gamma \vdash \alpha[n:N]t : \text{И}[n:N]T}\ (\text{И}\textsc{-intro}) \qquad \frac{\Gamma[n':N] \vdash (n:N) \# T \qquad \Gamma \vdash (n:N) \# t : \text{И}[n':N]T}{\Gamma \vdash t\ @\ n : \nu[n':N]\,(n\ \ n') \cdot T}\ (\text{И}\textsc{-elim})$$

Definitional freshness for types

$$\frac{\Gamma \vdash n : N \qquad \Gamma \vdash T \qquad \Gamma[n':N] \vdash (n\ \ n') \cdot T = T}{\Gamma \vdash (n:N) \# T}\ (\textsc{fresh-typ})$$

Definitional freshness for terms

$$\frac{\Gamma \vdash (n:N) \# T \qquad \Gamma[n':N] \vdash (n\ \ n') \cdot t = t : T}{\Gamma \vdash (n:N) \# t : T}\ (\textsc{fresh-term})$$

Definitionally equal types

$$\frac{\Gamma \vdash T \qquad \Gamma\Delta \vdash \mathrm{ok} \qquad [n:N], [n':N] \in \Delta}{\Gamma\Delta \vdash (n\ \ n') \cdot T = T}\ (\textsc{fresh-hyp-typ})$$

$$\frac{\Gamma[n:N] \vdash (n:N) \# T}{\Gamma[n:N] \vdash \nu[n:N]\,T = T}\ (\textsc{local-typ-comp})$$

(CONG-TYP): congruence properties for
$(n\ \ n') \cdot T$, $\nu[n:N]\,T$ and $\text{И}[n:N]T$ (omitted).

(SWAP-TYP-COMP): computational properties of swapping,
with well-formedness hypotheses omitted:

$$(n\ \ n)\cdot T = T$$
$$(n\ \ n')\cdot(n\ \ n')\cdot T = T$$
$$(n\ \ n')\cdot(m\ \ m')\cdot T = ((n\ \ n')m\ \ (n\ \ n')m')\cdot(n\ \ n')\cdot T$$

(where $(n\ \ n')m$ is n' if $m = n$, is n if $m = n'$ and is m otherwise)

$$(n\ \ n')\cdot N = N$$
$$(n\ \ n')\cdot\nu[n'' : N]\,T = \nu[n'' : N]\,(n\ \ n')\cdot T \text{ where } n'' \neq n, n'$$
$$(n\ \ n')\cdot \text{И}[n'' : N]T = \text{И}[n'' : N](n\ \ n')\cdot T \text{ where } n'' \neq n, n'$$
$$(n\ \ n')\cdot\Pi(x : T)T' = \Pi(x : (n\ \ n')\cdot T)\ (n\ \ n')\cdot T'((n\ \ n')\cdot x/x)$$

Definitionally equal terms

$$\frac{\Gamma \vdash t : T \qquad \Gamma\Delta \vdash \text{ok} \qquad [n : N], [n' : N] \in \Delta}{\Gamma\Delta \vdash (n\ \ n')\cdot t = t : T}\ (\text{FRESH-HYP-TERM})$$

$$\frac{\Gamma[n : N] \vdash (n : N)\ \#\ t : T}{\Gamma[n : N] \vdash \nu[n : N]\,t = t : T}\ (\text{LOCAL-TERM-COMP})$$

$$\frac{\Gamma[n' : N] \vdash (n : N)\ \#\ t : T \qquad n \neq n'}{\Gamma \vdash (\alpha[n' : N]t)\ @\ n = \nu[n' : N]\,(n\ \ n')\cdot t : \nu[n' : N]\,(n\ \ n')\cdot T}\ (\text{И-COMP})$$

$$\frac{\Gamma \vdash t : \text{И}[n : N]T \qquad n \notin \text{fn}(t)}{\Gamma \vdash t = \alpha[n : N](t\ @\ n) : \text{И}[n : N]T}\ (\text{И-UNIQ})$$

(CONG-TERM): congruence properties for
$(n\ \ n')\cdot t$, $\mathsf{if}\ t = n\ \mathsf{then}\ t'\ \mathsf{else}\ t''$, $\nu[n : N]\,t$, $\alpha[n : N]t$ and $t\ @\ n$ (omitted).

(SWAP-TERM-COMP): computational properties of swapping,
with well-formedness hypotheses and types omitted:

$$(n\ \ n)\cdot t = t$$
$$(n\ \ n')\cdot(n\ \ n')\cdot t = t$$
$$(n\ \ n')\cdot(m\ \ m')\cdot t = ((n\ \ n')m\ \ (n\ \ n')m')\cdot(n\ \ n')\cdot t$$

(where $(n\ \ n')m$ is n' if $m = n$, is n if $m = n'$ and is m otherwise)

$$(n\ \ n')\cdot n'' = (n\ \ n')n''$$
$$(n\ \ n')\cdot(\mathsf{if}\ t{=}n''\ \mathsf{then}\ t'\ \mathsf{else}\ t'')=\mathsf{if}\ (n\ \ n')\cdot t{=}(n\ \ n')n''\ \mathsf{then}\ (n\ \ n')\cdot t'\ \mathsf{else}\ (n\ \ n')\cdot t''$$
$$(n\ \ n')\cdot\nu[n'' : N]\,t = \nu[n'' : N]\,(n\ \ n')\cdot t \text{ where } n'' \neq n, n'$$
$$(n\ \ n')\cdot\alpha[n'' : N]t = \alpha[n'' : N](n\ \ n')\cdot t \text{ where } n'' \neq n, n'$$
$$(n\ \ n')\cdot\lambda(x : T)t = \lambda(x : (n\ \ n')\cdot T)\ (n\ \ n')\cdot t((n\ \ n')\cdot x/x)$$
$$(n\ \ n')\cdot t\,t' = ((n\ \ n')\cdot t)((n\ \ n')\cdot t')$$

(IF-COMP): computational properties of conditionals,
with well-formedness hypotheses and types omitted:

$$\mathsf{if}\ n = n\ \mathsf{then}\ t\ \mathsf{else}\ t' = t$$
$$\mathsf{if}\ n = n'\ \mathsf{then}\ t\ \mathsf{else}\ t' = t' \text{ (where } n \neq n')$$