Barendregt Convenes with Knaster and Tarski: Strong Rule Induction for Syntax with Bindings

JAN VAN BRÜGGE, Heriot-Watt University, United Kingdom
JAMES MCKINNA, Heriot-Watt University, United Kingdom
ANDREI POPESCU, University of Sheffield, United Kingdom
DMITRIY TRAYTEL, University of Copenhagen, Denmark

This paper is a contribution to the meta-theory of systems featuring syntax with bindings, such as λ-calculi and logics. It provides a general criterion that targets *inductively defined rule-based systems*, enabling for them inductive proofs that leverage *Barendregt's variable convention* of keeping the bound and free variables disjoint. It improves on the state of the art by (1) achieving high generality in the style of Knaster–Tarski fixed point definitions (as opposed to imposing syntactic formats), (2) capturing systems of interest without modifications, and (3) accommodating infinitary syntax and non-equivariant predicates.

CCS Concepts: • **Theory of computation** → **Logic and verification**.

Additional Key Words and Phrases: syntax with bindings, induction, formal reasoning, nominal sets

ACM Reference Format:
Jan van Brügge, James McKinna, Andrei Popescu, and Dmitriy Traytel. 2025. Barendregt Convenes with Knaster and Tarski: Strong Rule Induction for Syntax with Bindings. *Proc. ACM Program. Lang.* 9, POPL, Article 57 (January 2025), 32 pages. https://doi.org/10.1145/3704893

1 Introduction

Inductive definitions and proofs are a cornerstone of mathematics and theoretical computer science, and therefore solid and flexible foundations for induction are crucial in the development of these subjects—especially when it comes to the rigorous formulations and proofs of the results, using tools such as proof assistants. This paper is concerned with the formal foundations of induction for rule-based systems. Consider the basic example predicate describing whether a natural number is even:

$$\mathit{even}\ 0\ \text{(Base)} \qquad \frac{\mathit{even}\ n}{\mathit{even}\ (n+2)}\ \text{(Ind)}$$

The definition is inductive: a base case states that 0 is even, and an inductive case states that $n + 2$ is even if n is even. The intention is that all even numbers, and only those, are obtained by repeated application of the two rules; or equivalently, *even* is the smallest predicate closed under these rules.

One can take a *syntactic-format* approach to making sense of this and similar definitions, by proving a theorem such as: "For any specification consisting of rules where the conclusion and the hypotheses say that the to-be-defined inductive predicate applied to some arguments holds true, there exists the smallest predicate that satisfies the specification." Various relaxations and enhancements

Authors' Contact Information: Jan van Brügge, jsv2000@hw.ac.uk, Department of Computer Science, Heriot-Watt University, Edinburgh, United Kingdom; James McKinna, j.mckinna@hw.ac.uk, Department of Computer Science, Heriot-Watt University, Edinburgh, United Kingdom; Andrei Popescu, a.popescu@sheffield.ac.uk, School of Computer Science, University of Sheffield, Sheffield, United Kingdom; Dmitriy Traytel, traytel@di.ku.dk, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

ACM 2475-1421/2025/1-ART57
https://doi.org/10.1145/3704893

of such a format are possible, e.g., allowing non-recursive assumptions, side-conditions, and specifying a grammar for the arguments to which the to-be-defined predicate is applied. But no matter how far we go with format enhancements, we are likely to encounter situations where they are still not enough. Particularly difficult aspects to capture via formats are nested quantifiers and higher-order operators. For example, consider the set *Tree* of finite trees whose leaves *Leaf* are labelled by natural numbers and such that every tree $t \in \mathit{Tree}$ has a finite (possible empty) set $\mathit{Desc}\ t \in \mathcal{P}_{\mathsf{fin}}(\mathit{Tree})$ of immediate descendants. We can define inductively the following parity simulation relation $\preceq$ on trees:

$$\frac{t = \mathit{Leaf}\ n \qquad t' = \mathit{Leaf}\ (2 * n)}{t \preceq t'}\ \text{(Base)} \qquad \frac{\mathit{isDesc}\ t \qquad \mathit{isDesc}\ t' \qquad \mathit{RelSet}\ (\preceq)\ (\mathit{Desc}\ t)\ (\mathit{Desc}\ t')}{t \preceq t'}\ \text{(Ind)}$$

where *isDesc t* states that t is not a leaf and, for any binary relation R on a set A, *RelSet R* denotes its Hoare-style extension to a relation on $\mathcal{P}_{\mathsf{fin}}(A)$ defined by $\mathit{RelSet}\ R\ B\ B' = (\forall a \in B.\ \exists a' \in B'.\ R\ a\ a')$.

For making sense of rule-based inductive definitions, an approach that is more general and principled (and conceptually simpler!) than the syntactic-format approach is possible, by noticing that the existence of a smallest predicate satisfying a specification is guaranteed *regardless of its format*, provided it can be expressed using a monotonic operator on predicates. The operators underlying the definitions of *even* and $\preceq$ are G_{even} and $G_{\preceq}$ are defined as follows:

$G_{even}\ P\ m = (m = 0\ \vee\ \exists n.\ m = n + 2 \wedge P\ n)$

$G_{\preceq}\ R\ t\ t' = ((\exists n.\ t = \mathit{Leaf}\ n \wedge t' = \mathit{Leaf}\ (2{*}n)) \vee (\mathit{isDesc}\ t \wedge \mathit{isDesc}\ t' \wedge \mathit{RelSet}\ R\ (\mathit{Desc}\ t)\ (\mathit{Desc}\ t')))$

For any monotonic operator on a complete lattice (such as the lattice of predicates), as is easily seen to be the case with G_{even} and $G_{\preceq}$, the Knaster–Tarski theorem [Tarski 1955] guarantees the existence of a least fixed point. So *even* and $\preceq$ both exist as the least fixed points of G_{even} and $G_{\preceq}$, and have the desired properties, including induction principles for reasoning about them, merely by virtue of these operators being monotonic. The precise format of the predicate does not matter. In particular, for $\preceq$ the definition of *RelSet* is irrelevant, other than it is monotonic. This monotonicity-based approach was a major breakthrough, since it covers both existing and future syntactic formats that one would be interested in. It was implemented as part of the induction facilities of several proof assistants, notably the ones based on higher-order logic including HOL4 [Gordon and Melham 1993], HOL Light [Harrison 2024] and Isabelle/HOL [Nipkow et al. 2002].

Here we will be concerned with inductive definitions involving syntax with bindings—pervasive in the theory of logics and programming languages, where variables are being bound in terms and formulas via quantifiers, λ-abstractions, etc. When working with these systems, researchers want to avoid the overlap between bound and free variables, lest their proofs become significantly harder or fail altogether. This means applying Barendregt's famous *variable convention* [Barendregt 1985, p. 26]: "If [the terms] $M_1, \ldots, M_n$ occur in a certain mathematical context (e.g. definition, proof), then in these terms all bound variables are chosen to be different from the free variables."

This informal principle has been made rigorous by subsequent research, notably in the context of Nominal Logic [Gabbay and Pitts 1999, 2002] and related formalisms (e.g., [Aydemir et al. 2008]). Specifically for inductive rule-based systems involving binders, Urban et al. [2007] identified a rule format and some assumptions that are sufficient for allowing Barendregt's variable convention to be soundly used in proofs, leading to a *strong induction principle* criterion guaranteeing the disjointness between bound and free variables. Subsequently, this criterion has been implemented as part of the Nominal Isabelle package [Urban 2008; Urban and Kaliszyk 2012].

A natural question to ask is whether **(1)** a more general, monotonicity-based approach (that does not require a syntactic format) can be pursued here. Besides the limitations stemming from the syntactic format, another limitation of the state of the art is that **(2)** it fails to directly capture existing mainstream systems such as the λ-calculus reduction and π-calculus transition relations, but requires the modification of these systems' standard presentations by adding more side-conditions. In other words, there is a gap between the standard definitions of these systems from textbooks and

the formal requirements for enabling the variable convention. Finally, the state of the art, deeply rooted in Nominal Logic and its finite support and equivariance conditions (the latter expressing a form of uniform behavior of functions and predicates) [Gabbay and Pitts 2002; Pitts 2006], **(3)** does not cover infinitary syntax with bindings such as infinitary extensions of the λ-calculus [Barendregt and Klop 2009; Mazza 2012] and first-order logic [Hanf 1964; Makkai 1969].

This paper makes contributions along all the above three axes. It introduces general criteria for when inductive systems are variable-convention observing, leveraging monotonicity and Knaster–Tarski. Our criteria also fill the aforementioned formality gap as they apply to the systems without modifications, and moreover cope with infinitary syntax and the lack of equivariance.

Overview. We start with revisiting standard examples coming from the λ-calculus (§2), highlighting the limitations of the state of the art. Then, after recalling the necessary background concepts pertaining to nominal sets and induction (§3), we prove the initial version of our main result, a format-free general criterion for strong rule induction (§4). This initial version will be further improved and generalized throughout the rest of the paper by challenging it with inductive systems whose syntactic structures or binding dynamics are increasingly sophisticated. We first deploy our criterion to tackle the motivating examples (§5), which leads us to a deployment heuristic (§6). We compare our criterion with the state of the art criterion of Urban et al. [2007] with respect to the addition of side-conditions (§7). More examples are discussed (§8), including the π-calculus and subtyping for System $F_{<:}$, the latter suggesting a strengthening of our criterion with inductive information. Further examples take us into the realm of infinitary structures with bindings (§9), such as extensions of first-order logic that allow infinitary cardinal-bounded conjunctions and quantifications in formulas (§9.1). To extend our criterion for coping with predicates defined over infinitary structures, we introduce what we call *loosely-supported nominal sets* (§9.2), a variation of nominal sets equipped with a "loose" (not necessarily minimal) supporting set operator that relax the finite-support assumption to a small-support one, where "small" is understood with respect to a given infinite cardinal. The last example we consider involves the meta-theory of an affine infinitary λ-calculus (§9.3), and leads to a further generalization of our criterion to handle non-equivariant predicates (§9.4). We describe a tool that we have implemented in Isabelle to support our formalization of the general theory and the examples, as well as case studies based on these examples (§10), and conclude with more related work (§11). Our technical report [van Brügge et al. 2025b] gives more details about this paper's constructions and results, and our Isabelle mechanization.

2 Motivating Example: λ-Calculus

In this section, *Var*, the set of variables, will be a countably infinite set. We consider the syntax of the (untyped) λ-calculus, defining the set *LTerm* of *λ-terms*, ranged over by t, s etc., via the grammar:

$$t \ ::= \ Vr\ x \ \mid \ Ap\ t_1\ t_2 \ \mid \ Lm\ x\ t$$

Thus, a λ-term is either (the injection of) a variable, or an application, or a λ-abstraction of a variable in a term. We also assume that, in a term of the form $Lm\ x\ t$, the variable x is bound in t; and terms are equated modulo the induced notion of alpha-equivalence, e.g., $Lm\ x\ (Vr\ x) = Lm\ y\ (Vr\ y)$. For any λ-term t, we write $FV\ t$ for its set of *free variables*. A variable x is *fresh* for t when $x \notin FV\ t$. We write $t[s/x]$ for the (capture-avoiding) *substitution* of the term s for the variable x in the term t.

A fundamental relation on this syntax is *β-reduction*, the binary relation $\Rightarrow$ between λ-terms defined in Fig. 1. If we ignore binding information, the standard proof principle associated to this definition is the following *rule induction* principle:

Prop 1. Let $\varphi : LTerm \longrightarrow LTerm \longrightarrow Bool$ and assume that:

- (⟦Beta⟧): $\forall x, t_1, t_2.\ \varphi\ (Ap\ (Lm\ x\ t_1)\ t_2)\ (t_1[t_2/x])$

$$Ap\ (Lm\ x\ t_1)\ t_2 \Rightarrow t_1[t_2/x]\ \text{(Beta)} \qquad \frac{t \Rightarrow t'}{Lm\ x\ t \Rightarrow Lm\ x\ t'}\ \text{(Xi)}$$

$$\frac{t_1 \Rightarrow t_1'}{Ap\ t_1\ t_2 \Rightarrow Ap\ t_1'\ t_2}\ \text{(ApL)} \qquad \frac{t_2 \Rightarrow t_2'}{Ap\ t_1\ t_2 \Rightarrow Ap\ t_1\ t_2'}\ \text{(ApR)}$$

Fig. 1. λ-calculus β-reduction

- (⟪Xi⟫): $\forall x, t, t'.\ ((t \Rightarrow t') \wedge \varphi\ t\ t') \longrightarrow \varphi\ (Lm\ x\ t)\ (Lm\ x\ t')$
- (⟪ApL⟫): $\forall t_1, t_1', t_2.\ \big((t_1 \Rightarrow t_1') \wedge \varphi\ t_1\ t_1'\big) \longrightarrow \varphi\ (Ap\ t_1\ t_2)\ (Ap\ t_1'\ t_2)$
- (⟪ApR⟫): $\forall t_1, t_2, t_2'.\ \big((t_2 \Rightarrow t_2') \wedge \varphi\ t_2\ t_2'\big) \longrightarrow \varphi\ (Ap\ t_1\ t_2)\ (Ap\ t_1\ t_2')$

Then $\forall t, t'.\ (t \Rightarrow t') \longrightarrow \varphi\ t\ t'$.

Thus, standard induction allows us to infer that $\Rightarrow$ is included in a relation φ provided φ is closed under the rules defining $\Rightarrow$, i.e., uses that $\Rightarrow$ is the smallest relation closed under these rules.

However, due to the presence of bindings, it is desirable to have a stronger induction proof principle—featuring an enhancement that formalizes Barendregt's variable convention. For example, say we want to prove that β-reduction is closed under substitution, i.e, $(t \Rightarrow t') \longrightarrow (t[s/y] \Rightarrow t'[s/y])$. The proof would go by rule induction, taking $\varphi\ t\ t'$ to be $\forall s, y.\ t[s/y] \Rightarrow t'[s/y]$. In the (⟪Beta⟫) case we must prove $\varphi\ (Ap\ (Lm\ x\ t_1)\ t_2)\ (t_1[t_2/x])$, i.e., for all s, y,

$$\textbf{(i)}\ \ (Ap\ (Lm\ x\ t_1)\ t_2)[s/y] \Rightarrow t_1[t_2/x][s/y]$$

To continue, we wish to move the $\_[s/y]$ substitution inside the constructors Ap and Lm on the left, and also inside the $t_1[t_2/x]$ substitution on the right, thus reducing the above to

$$\textbf{(ii)}\ \ Ap\,(Lm\,x\,(t_1[s/y]))\,(t_2[s/y]) \Rightarrow (t_1[s/y])[(t_2[s/y])/x]$$

the last being provable as an instance of the (Beta) rule, taking t_1 and t_2 from the rule to be $t_1[s/y]$ and $t_2[s/y]$. (Without being able to perform the above "moves", the proof would become quite complicated, as the goal would need to be generalized to work inductively.)

However, while substitution can soundly be moved inside applications (since by definition it commutes with applications), it is not always sound to move it inside λ-abstractions or other substitutions, *unless certain side-conditions hold.* In this case, we would need that *x is fresh for the parameters*, i.e., x is fresh for s and is different from y, which would ensure $(Lm\ x\ t_1)[s/y] = Lm\ x\ (t_1[s/y])$ and $t_1[t_2/x][s/y] = (t_1[s/y])[(t_2[s/y])/x]$, making (i) reducible to (ii) as desired for finishing the proof in the (⟪Beta⟫) case. Barendregt's insight, expressed in his variable convention and deployed systematically in proofs all throughout his λ-calculus monograph [Barendregt 1985], was that such freshness assumptions are usually safe, in that they do not lose generality (hence do not lead to incorrect reasoning).

Yet, Barendregt did not indicate exactly *when*, or *why*, such assumptions are safe. A rigorous answer to these questions was provided by Urban et al. [2007] (having prior roots in McKinna and Pollack [1999]; Pitts [2003]) who formalized the variable convention used in proof contexts like the above as a *strong rule induction* that allows assuming the rules' bound variables (e.g., x in (Beta) and (Xi)) to be fresh for given parameters (e.g., s and y). Here is the desired strong rule induction for β-reduction, where $\mathcal{P}_{\text{fin}}(Var)$ is the set of finite sets of variables:

Prop 2. Let P be a set of items called *parameters* and $Psupp : P \longrightarrow \mathcal{P}_{\text{fin}}(Var)$. Let $\varphi : P \longrightarrow LTerm \longrightarrow LTerm \longrightarrow Bool$ and assume that:

- (⟪Beta⟫): $\forall p, x, t_1, t_2.\ x \notin Psupp\ p \longrightarrow \varphi\ p\ (Ap\ (Lm\ x\ t_1)\ t_2)\ (t_1[t_2/x])$
- (⟪Xi⟫): $\forall p, x, t, t'.\ x \notin Psupp\ p \wedge (t \Rightarrow t') \wedge (\forall q.\ \varphi\ q\ t\ t') \longrightarrow \varphi\ p\ (Lm\ x\ t)\ (Lm\ x\ t')$
- (⟪ApL⟫): $\forall p, t_1, t_1', t_2.\ (t_1 \Rightarrow t_1') \wedge (\forall q.\ \varphi\ q\ t_1\ t_1') \longrightarrow \varphi\ p\ (Ap\ t_1\ t_2)\ (Ap\ t_1'\ t_2)$
- (⟪ApR⟫): $\forall p, t_1, t_2, t_2'.\ (t_2 \Rightarrow t_2') \wedge (\forall q.\ \varphi\ q\ t_2\ t_2') \longrightarrow \varphi\ p\ (Ap\ t_1\ t_2)\ (Ap\ t_1\ t_2')$

Then $\forall p, t, t'.\ (t \Rightarrow t') \longrightarrow \varphi\ p\ t\ t'$.

The predicate to be proved is now quantified universally over parameters, whose role is to provide the variables that one would like to avoid within inductive proofs—what Barendregt's convention, cited in the introduction, calls "the free variables" in a "certain mathematical context". To use this principle in the proof discussed above, we take P to be $LTerm \times Var$ and $Psupp(s, y)$ to be $FV\ s \cup \{y\}$. (Note that a weaker form of this principle would fix a parameters p rather than quantifying universally over parameters, so that φ would not have a parameter argument and, for example, the hypothesis (⦇Xi⦈) would become $\forall x, t, t'.\ x \notin Psupp\ p \wedge (t \Rightarrow t') \wedge \varphi\ t\ t' \longrightarrow \varphi\ (Lm\ x\ t)\ (Lm\ x\ t')$ and (⦇ApL⦈) and (⦇ApL⦈) would become the usual inductive conditions from the standard rule induction expressed by Prop. 1. While often the fixed-parameter version is good enough, as is the case with the proof discussed above which works for fixed s and y, sometimes the extra flexibility of quantifying universally over parameters is important—Lemma 107 from App. F of our technical report [van Brügge et al. 2025b] (reflexivity of the System $F_{<:}$ typing from POPLmark 1A [Aydemir et al. 2005]) gives an example for structural induction which is a particular case of rule induction.)

Importantly, Urban et al. [2007] have also noted that Barendregt's variable convention is not sound for all inductively defined relations on λ-terms, and have provided a syntactic criterion for when it *is* sound. Unfortunately, their criterion does not cover the above (standard) definition of β-reduction (shown in Fig. 1) but only a modification of it obtained by adding a freshness side-condition to the (Beta) rule:

$$\frac{}{Ap\ (Lm\ x\ t_1)\ t_2 \Rightarrow t_1[t_2/x]}\ \begin{array}{l}\text{(Beta')}\\ [x \notin FV\ t_2]\end{array}$$

With this modification, strong induction for β-reduction, i.e., Prop. 2, becomes an instance of their syntactic criterion. This variant of β-reduction, with (Beta') instead of (Beta), can be proved equivalent to the standard one, but this is far from immediate.

The need for adding side-conditions arises quite pervasively when instantiating Urban et al.'s result to examples. In fact, the authors themselves show such an example in their paper: a parallel β-reduction [Lévy 1975; Takahashi 1995], where they must change the "Parallel Beta" rule

$$\frac{t_1 \Longrightarrow t_1' \qquad t_2 \Longrightarrow t_2'}{Ap\ (Lm\ x\ t_1)\ t_2 \Longrightarrow t_1'[t_2'/x]}\ \text{(ParBeta)}$$

into the weaker rule

$$\frac{t_1 \Longrightarrow t_1' \qquad t_2 \Longrightarrow t_2'}{Ap\ (Lm\ x\ t_1)\ t_2 \Longrightarrow t_1'[t_2'/x]}\ \begin{array}{l}\text{(ParBeta')}\\ [x \notin FV\ t_2 \cup FV\ t_2']\end{array}$$

Quoting from Urban et al.: "This is annoying because both versions can be shown to define the same relation, but we have no general, and automatable, method for determining this."

Another limitation of their criterion (again acknowledged by the authors themselves) is its syntactic-format nature, requiring rules the form

$$\frac{\varphi\ p_1\ \vec{s_1} \quad \ldots \quad \varphi\ p_n\ \vec{s_n}}{\varphi\ p\ \vec{t}}\ [\text{side-conditions}]$$

which is quite rigid. In particular this forbids, in the rules' assumptions, the occurrence of the defined relation under universal or existential quantifiers, or under other higher-order operators. Our results will lift both of the above limitations.

3 Preliminaries on Nominal Sets and Knaster–Tarski Fixpoints

Next we recall some background on nominal sets [Gabbay and Pitts 2002; Pitts 2013] and induction based on Knaster–Tarski fixpoints [Knaster 1928; Tarski 1955].

Nominal Sets. Let *Var* be a fixed countable sets of items called *variables*, or *atoms*. Given any function $f : Var \to Var$, the *core* of f is defined as the set of all variables that are changed by f: $Core\ f = \{x \mid f\ x \neq x\}$. (What we call "core" is usually called the "support" of f, which is consistent with the more general notion of support we discuss next. But we prefer to name it differently because of its bootstrapping role towards the general notion.) Let *Perm*, ranged over by σ, denote the set of (finite) *permutations*, i.e., bijections on *Var* of finite core.

Note that $(Perm, \circ, 1_{Var})$ forms a group, where 1_{Var} is the identity permutation and $\circ$ is composition. A *pre-nominal set* is a set equipped with a *Perm*-action, i.e., a pair $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$ where A is a set and $\_[\_]^{\mathcal{A}} : A \to Perm \to A$ is an action of the monoid *Perm* on A, in that it is idle for identity ($a[1_{Var}]^{\mathcal{A}} = a$ for all $a \in A$) and compositional ($a[\sigma \circ \tau]^{\mathcal{A}} = a[\tau]^{\mathcal{A}}[\sigma]^{\mathcal{A}}$). Given $\sigma \in Perm$, we sometimes write $\_[\sigma]$ for the function in $A \to A$ (which is actually a bijection) that applies this fixed permutation. We let $Im\ \sigma$ be the operator in $\mathcal{P}(Var) \to \mathcal{P}(Var)$ that takes any $X \subseteq Var$ to the image of X through σ, namely $Im\ \sigma\ X = \{\sigma\ x \mid x \in X\}$. We write $x \leftrightarrow y$ for the permutation that takes x to y, y to x and all other variables to themselves; applying these permutations (to elements of a nominal set) will be called *swapping*.

Given a pre-nominal set $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$, an $a \in A$ and a set $X \subseteq Var$, we say that a *is supported by* X, or X *supports* a, if $a[x \leftrightarrow y]^{\mathcal{A}} = a$ holds for all $x, y \in Var \setminus X$, or equivalently, if $a[\sigma]^{\mathcal{A}} = a$ holds for all $\sigma \in Perm$ such that $\forall x \in X.\ \sigma\ x = x$. An element $a \in A$ is called *finitely supported* if there exists a finite set X that supports a. A *nominal set* is a pre-nominal set where every element is finitely supported. If $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$ is a nominal set and $a \in A$, then the smallest set that supports a can be shown to exist—it is denoted by $Supp^{\mathcal{A}}\ a$ and called the *support of* a.

Given two pre-nominal sets $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$ and $\mathcal{B} = (B, \_[\_]^{\mathcal{B}})$, the set $F = (A \to B)$ of functions from A to B naturally forms a pre-nominal set $\mathcal{F} = (F, \_[\_]^{\mathcal{F}})$ by defining $f[\sigma]$ to be the function that sends each $a \in A$ to $f(a[\sigma^{-1}])[\sigma]$. (So in particular we can talk about the notion of a set of variables supporting such a function.) $\mathcal{F}$ is not a nominal set, because not all functions are finitely supported, but we obtain a nominal set if we restrict it to the finitely supported functions. In addition to the above function-space construction, nominal set structures can also be naturally defined on the products, sums, container-type extensions (such as lists or trees) and quotients of the carrier sets, overall enjoying good category-theoretic properties, in particular forming a topos equivalent to the Schanuel topos [Pitts 2013].

The set of λ-terms with their standard *Perm*-action, $(LTerm, \_[\_])$, forms a nominal set, where the support of a term t consists of its free variables. Note that set $FV\ t$ of free variables of a λ-term t is traditionally defined recursively on the structure of t and not from permutation like the support is. However, writing *Supp* for the support operator of the nominal set $(LTerm, \_[\_])$, it can be checked that (1) t is supported by $FV\ t$ in that $t[x \leftrightarrow y] = t$ holds whenever $x, y \notin FV\ t$, by an easy induction on t; and that (2) for any x, assuming $x \in FV\ t \setminus Supp\ t$ yields a contradiction by taking some $y \notin FV\ t \cup Supp\ t$ and noting that $t[x \leftrightarrow y] \neq t$ (which again follows by easy induction from $x \in FV\ t$ and $y \notin FV\ t$) contradicts the fact that $x, y \notin Supp\ t$. Points (1) and (2) make $FV\ t$ coincide with $Supp\ t$. It is known (and can be established by an argument similar to the one sketched above) that this coincidence between the free-variable operator and support holds for all syntaxes with statically scoped bindings [Pitts 2006], so any such syntax forms a nominal set where the support is given by the free variables.

Although the concept of nominal set abstracts away from, and goes beyond syntactic objects (covering for example restricted spaces of functions, of semantic entities etc. [Pitts 2013]), it is often useful to think of the elements a of a nominal set as "term-like" entities; in this spirit, we will refer to the elements of $Supp^{\mathcal{A}}\ a$ as the free-variables of a.

Nominal sets underpin the semantics of *nominal logic* [Gabbay and Pitts 1999, 2002], a successful foundation tailored for reasoning about syntax with bindings. But nominal sets and nominal logic

techniques can also be used from within general-purpose foundations such as higher-order logic [Pitts 2006; Urban and Tasson 2005]—in this paper we subscribe to this approach.

Central in nominal logic, and in our own developments as well, is the notion of equivariance, which for a function, predicate or assertion means commutation with permutation actions. With roots in classical algebra [Pitts 2013, §1.1], equivariance has the following intuition in the context of syntax with bindings, as explained in the seminal nominal logic paper [Gabbay and Pitts 1999, §2]: "Properties of syntax should be sensitive only to distinctions between variable names, rather than to the particular names themselves." Here is the formal definition:

Def 3. Given two pre-nominal sets $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$ and $\mathcal{B} = (B, \_[\_]^{\mathcal{B}})$, a function $F : A \to B$ between their carrier sets is called *equivariant* when it commutes with the permutation actions: $F\ (a[\sigma]^{\mathcal{A}}) = (F\ a)[\sigma]^{\mathcal{B}}$ for all $a \in A$ and $\sigma \in \mathit{Perm}$.

Since the two-element set of Booleans (like any set) can be trivially equipped with identity permutation action to become a (pre-)nominal set, we can speak of the equivariance of predicates, $\varphi : A \to \mathit{Bool}$ where $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$ is a pre-nominal set. Here, equivariance can be equivalently expressed using implication: $\varphi\ a \longrightarrow \varphi\ (a[\sigma]^{\mathcal{A}})$ for all $a \in A$ and $\sigma \in \mathit{Perm}$.

The Knaster–Tarski Fixpoint Theorem. This celebrated result offers a simple yet powerful foundation for induction, with applications in areas such as semantics, verification and static analysis.

Thm 4. [Tarski 1955] Let $(L, \leq)$ be a complete lattice and $G : L \to L$ a monotonic operator. Then there exists a (unique) least fixpoint I_G for G, in that: $G\ I_G = I_G$ and $\forall k \in G.\ G\ k = k \longrightarrow I_G \leq k$. And I_G is the least pre-fixpoint as well, in that $\forall k \in L.\ G\ k \leq k \longrightarrow I_G \leq k$; finally, a practically useful variation of this also holds, where $\wedge$ is binary infimum in L: $\forall k \in L.\ G\ (I_G \wedge k) \leq k \longrightarrow I_G \leq k$.

It is the "pre-fixpoint" part of this theorem that enables inductive reasoning: To prove that $I_G \leq k$, it suffices to prove that $G\ (I_G \wedge k) \leq k$. While the theorem works in general for complete lattices, we will only use it for the particular lattices of predicates (equivalently, lattices of subsets), as initially formulated by Knaster [1928]. Given a set A and two predicates $\varphi, \psi : A \to \mathit{Bool}$ on it, we define $\varphi \leq \psi$ to be component-wise implication, namely $\forall a \in A.\ \varphi\ a \longrightarrow \psi\ a$. And indeed, $\leq$ is a complete-lattice order on the set $A \to \mathit{Bool}$ of predicates. This applies to n-ary predicates as well if we take A to be a product $A_1 \times \ldots \times A_n$. Often the operator G on predicates is given by a set of rules, and for this reason the emerging induction principle associated to I_G is referred to as *rule induction*.

4 Strong Rule Induction Criterion

Our main result, which we present next (Thm. 7 below), is an extension of Knaster–Tarski based rule induction to strong (variable-convention observing) induction, leveraging nominal-set structure.

We start with a monotonic operator $G : (T \to \mathit{Bool}) \to (\mathcal{P}_{\mathsf{fin}}(\mathit{Var}) \to T \to \mathit{Bool})$, where monotonicity again refers to the standard predicate orderings (component-wise implication). We iterate G to define the predicate $I_G : T \to \mathit{Bool}$ inductively as follows: $\dfrac{G\ I_G\ B\ t}{I_G\ t}$

We think of the above as the inductive specification of a rule-based system I_G. But differently from the usual Knaster–Tarski setting for such specifications, here we have made explicit an additional "bound variable set" argument $B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$ for the predicate returned by G. Our strong rule induction criterion will make assumptions on, and draw conclusions from, how G operates B.

But first let us make sense of the above specification of I_G without treating B specially. That I_G was obtained by "iterating" G means that I_G is the least (pre-)fixpoint of the operator $\lambda\, \varphi.\ \lambda\, t \in T.\ \exists B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var}).\ G\ \varphi\ B\ t$. Its existence is guaranteed by Thm. 4, taking I_G to be the least fixpoint of the operator on $(T \to \mathit{Bool}) \to (T \to \mathit{Bool})$ that acts like G but applies existential quantification

over B, i.e., sends any predicate $\varphi : T \to \mathit{Bool}$ to $\lambda t.\ \exists B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var}).\ G\ \varphi\ B\ t$. The standard rule induction principle stemming from I_G's definition (via Thm. 4) is the following:

Thm 5. Assume G is monotonic. If $\varphi : T \to \mathit{Bool}$ is such that $\forall t \in T.\ (\exists B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var}).\ G\ (\lambda t'.\ I_G\ t' \wedge \varphi\ t')\ B\ t) \longrightarrow \varphi\ t$, then $I_G \leq \varphi$, i.e., $\forall t \in T.\ I_G\ t \longrightarrow \varphi\ t$.

(The assumption of Thm. 5 is equivalent to $\forall t \in T, \forall B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var}).\ G\ (\lambda t'.\ I_G\ t' \wedge \varphi\ t')\ B\ t \longrightarrow \varphi\ t$.)

Now let us make our move towards strong rule induction. To formulate such a principle without knowing how G looks like, we think of G as the rules defining our predicate I_G; and of its argument B as the bound variables appearing in the *conclusions* of these rules—for this interpretation to make sense, we assume that I_G operates on "term-like" entities, i.e., elements of a nominal set $\mathcal{T}$. Our key observation is that Barendregt's variable convention rests on the bound variables being "refreshable" in the rules, in that (roughly speaking) we can always rename them to become fresh for a rule's *entire* conclusion (and not just the location where they are bound) *without invalidating its hypotheses*. To model this, we introduce the concept of $\mathcal{T}$-refreshability; and also introduce $\mathcal{T}$-freshness, which goes further to say that the bound variables are already fresh.

Def 6. Given a nominal set $\mathcal{T} = (T, \_[\_]^{\mathcal{T}})$, an operator $G : (T \to \mathit{Bool}) \to (\mathcal{P}_{\mathsf{fin}}(\mathit{Var}) \to T \to \mathit{Bool})$ is said to be:
- *$\mathcal{T}$-refreshable* when, for all $\varphi : T \to \mathit{Bool}$, $B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$ and $t \in T$, if φ is equivariant and $G\ \varphi\ B\ t$ then there exists $B' \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$ such that $B' \cap \mathit{Supp}^{\mathcal{T}}\ t = \emptyset$ and $G\ \varphi\ B'\ t$;
- *$\mathcal{T}$-fresh* when, for all $\varphi : T \to \mathit{Bool}$, $B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$ and $t \in T$, if $G\ \varphi\ B\ t$ then $B \cap \mathit{Supp}^{\mathcal{T}}\ t = \emptyset$.

(Note that $\mathcal{T}$-freshness implies $\mathcal{T}$-refreshability, taking $B' = B$.)

And indeed, we can prove that $\mathcal{T}$-refreshability in conjunction with equivariance (which essentially ensures robustness of the rules in the refreshing process) is sufficient for enabling strong rule induction. In what follows, a pair (P, Psupp) where P is a set and $\mathit{Psupp} : P \to \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$ will be called *parameter structure*. (These are not required to be nominal sets.)

Thm 7. Let $\mathcal{T} = (T, \_[\_]^{\mathcal{T}})$ be a nominal set and $G : (T \to \mathit{Bool}) \to (\mathcal{P}_{\mathsf{fin}}(\mathit{Var}) \to T \to \mathit{Bool})$ a monotonic, equivariant and $\mathcal{T}$-refreshable operator. Let (P, Psupp) be a parameter structure and $\varphi : P \to T \to \mathit{Bool}$ a predicate. Assume that:

$$\forall p \in P, t \in T, B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var}).\ \left(\begin{array}{c} B \cap (\mathit{Psupp}\ p \cup \mathit{Supp}^{\mathcal{T}}\ t) = \emptyset\ \wedge \\ G\ (\lambda t'.\ I_G\ t' \wedge \forall p' \in P.\ \varphi\ p'\ t')\ B\ t \end{array} \right) \longrightarrow \varphi\ p\ t$$

Then $\forall p \in P.\ I_G \leq \varphi\ p$, i.e., $\forall p \in P, t \in T.\ I_G\ t \longrightarrow \varphi\ p\ t$.

Highlighted above is the "strength" of the stated strong induction principle for I_G: When performing induction, we are allowed to assume the variables of the parameter p, and also the free variables of the nominal-set (i.e., the term-like entity) argument t, to be distinct from the variables in B (the bound variables). In short, the bound variables can be avoided.

We show a detailed proof of this result, partly because we will later do a bit of proof mining for generalizing it. The main idea is that, using $\mathcal{T}$-refreshability, we are able to "clean up" the inductive definition of I_G to assume freshness of the bound-variables B for the rules' conclusions t (i.e., $B \cap \mathit{Supp}^{\mathcal{T}}\ t = \emptyset$), and then use G's equivariance to prove that freshness for the parameters can also be assumed.

Proof. We will write $\_[\_]$ instead of $\_[\_]^{\mathcal{T}}$ and *Supp* instead of $\mathit{Supp}^{\mathcal{T}}$.

We first define an inductive predicate I'_G which is a variation of I_G that factors in "half" of the intended freshness assumption, namely $B \cap \mathit{Supp}\ t = \emptyset$:

$$\frac{G\ I'_G\ B\ t \qquad B \cap \mathit{Supp}\ t = \emptyset}{I'_G\ t}$$

Since the defining rule for I'_G is weaker (has more hypotheses), I'_G is stronger than I_G, we have: **(1)** $\forall t.\ I'_G\ t \longrightarrow I_G\ t$. Crucially, we will be able to also prove the converse of (1). But first we need: **(2)** I'_G is equivariant, i.e., $\forall \sigma \in \mathit{Perm},\ t \in T.\ I'_G\ t \longrightarrow I'_G\ (t[\sigma])$.

The proof of (2) goes by rule induction on the definition of I'_G, for an (arbitrary but) fixed $\sigma \in \mathit{Perm}$: We fix $B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$ and $t \in T$ and assume **(i)** $G\ (I'_G \circ (\_[\sigma]))\ B\ t$ and **(ii)** $B \cap \mathit{Supp}\ t = \emptyset$. We must show that $I'_G\ (t[\sigma])$. Using the introduction rule associated to the definition of I'_G, it suffices to show **(i')** $G\ I'_G\ (\mathit{Im}\ \sigma\ B)\ (t[\sigma])$ and **(ii')** $\mathit{Im}\ \sigma\ B \cap \mathit{Supp}\ (t[\sigma]) = \emptyset$. From (i) and the equivariance of G, we obtain $G\ (I'_G \circ (\_[\sigma]) \circ (\_[\sigma^{-1}])\ (\mathit{Im}\ \sigma\ B)\ (t[\sigma])$, hence, by the fact that $\sigma \circ \sigma^{-1} = 1_{\mathit{Var}}$ and the functoriality of $\_[\_]$, we obtain (i'), as desired. Moreover, (ii') follows from (ii) and the properties of *Supp*. (Note that so far we used G's equivariance and monotonicity, but not yet its $\mathcal{T}$-refreshability.)

Now we prove **(3)** $\forall t.\ I_G\ t \longrightarrow I'_G\ t$, by rule induction on the definition of I_G: We fix $B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$ and $t \in T$ and assume (iii) $G\ I'_G\ B\ t$. We must show $I'_G\ t$. From (2), (iii) and $\mathcal{T}$-refreshability, we obtain $B' \in \mathcal{P}_{\mathsf{fin}}\ (\mathit{Var})$ such that $B' \cap \mathit{Supp}\ t = \emptyset$ and $G\ I'_G\ B'\ t$. Hence, $I'_G\ t$ follows by I'_G's introduction rule.

From (1) and (3), we have **(4)** $I_G = I'_G$. Now we are ready to tackle the theorem's statement, in which, using (4), we will freely replace I_G with I'_G. Thus, we assume

$$\textbf{(5)}\ \forall p \in P, t \in T, B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var}).\ \left(\begin{array}{c} B \cap (\mathit{Psupp}\ p \cup \mathit{Supp}\ t) = \emptyset\ \wedge \\ G\ (\lambda t'.\ I'_G\ t' \wedge \forall p' \in P.\ \varphi\ p'\ t')\ B\ t \end{array}\right) \longrightarrow \varphi\ p\ t$$

We must prove $\forall p \in P, t \in T.\ I'_G\ t \longrightarrow \varphi\ p\ t$, i.e., $\forall t \in T.\ I'_G\ t \longrightarrow (\forall p \in P.\ \varphi\ p\ t)$. We will prove something more general, namely that I'_G implies the equivariant envelope of φ: $\forall t \in T.\ I'_G\ t \longrightarrow (\forall \sigma \in \mathit{Perm}.\ \forall p \in P.\ \varphi\ p\ (t[\sigma]))$.

We again proceed by rule induction on the definition of I'_G: We fix $B \in \mathcal{P}_{\mathsf{fin}}(\mathit{Var})$, $t \in T$, $\sigma \in \mathit{Perm}$ and $p \in P$ and assume **(iv)** $G\ (\lambda t'.\ I'_G\ t' \wedge (\forall \sigma' \in \mathit{Perm},\ p' \in P.\ \varphi\ p'\ (t'[\sigma'])))\ B\ t$ and **(v)** $B \cap \mathit{Supp}\ t = \emptyset$. We must show $\varphi\ p\ (t[\sigma])$.

Let $B' = \mathit{Im}\ \sigma\ B$. Note that B' is finite because B is. From (v) and the properties of *Supp*, we have **(v')** $B' \cap \mathit{Supp}\ (t[\sigma]) = \emptyset$.

Note that $\mathit{Psupp}\ p \cup \mathit{Supp}\ (t[\sigma])$ is finite because both $\mathit{Psupp}\ p$ and $\mathit{Supp}\ (t[\sigma])$ are finite. With the finiteness of B' and (v'), we obtain the existence of $\tau \in \mathit{Perm}$ such that

(vi) $\mathit{Im}\ \tau\ B' \cap (\mathit{Psupp}\ p \cup \mathit{Supp}\ (t[\sigma])) = \emptyset$ and **(vii)** $\forall x \in \mathit{Supp}\ (t[\sigma]).\ \tau\ x = x$.

Let $\delta = \tau \circ \sigma$. By the functoriality of $\_[\_]$, we have $t[\delta] = t[\sigma][\tau]$. Also, from (vii) and the properties of *Supp*, we have $t[\sigma][\tau] = t[\sigma]$. Hence **(viii)** $t[\delta] = t[\sigma]$. Note also that, by the definitions of δ and B' we have **(ix)** $\mathit{Im}\ \delta\ B = \mathit{Im}\ \tau\ B'$.

From (iv) and the monotonicity of G, we have $G\ (\lambda t'.\ I'_G\ t' \wedge (\forall p' \in P.\ \varphi\ p'\ (t'[\delta])))\ B\ t$.
Hence, by G's monotonicity and I'_G's equivariance, $G\ (\lambda t'.\ I'_G\ (t'[\delta]) \wedge (\forall p' \in P.\ \varphi\ p'\ (t'[\delta])))\ B\ t$.
Hence, by G's equivariance, $G\ (\lambda t'.\ I'_G\ (t'[\delta^{-1}][\delta]) \wedge (\forall p' \in P.\ \varphi\ p'\ (t'[\delta^{-1}][\delta])))\ (\mathit{Im}\ \delta\ B)\ (t[\delta])$.
Hence, by $\_[\_]$'s functoriality and $\delta \circ \delta^{-1} = 1_{\mathit{Var}}$, $G\ (\lambda t'.\ I'_G\ t' \wedge (\forall p' \in P.\ \varphi\ p'\ t'))\ (\mathit{Im}\ \delta\ B)\ (t[\delta])$.
Hence, using (viii) and (ix), $G\ (\lambda t'.\ I'_G\ t' \wedge (\forall p' \in P.\ \varphi\ p'\ t'))\ (\mathit{Im}\ \tau\ B')\ (t[\sigma])$.
From this, (vi) and (5), we get $\varphi\ p\ (t[\sigma])$, as desired. □

5 The Motivating Example Revisited

Thm. 7 generalizes Prop. 2, and is in relation to Thm. 5 what Prop. 2 is in relation to Prop. 1, where β-reduction is generalized to an arbitrary inductively defined predicate I_G on a nominal set. Indeed, we obtain Prop. 2 by instantiating, in Thm. 7, $\mathcal{T}$ to the canonical nominal-set structure on LTerm^2 and $G : (\mathit{LTerm}^2 \to \mathit{Bool}) \to (\mathcal{P}_{\mathsf{fin}}(\mathit{Var}) \to \mathit{LTerm}^2 \to \mathit{Bool})$ to the operator described in Fig. 2.

Remark 8. In fact, instantiating Thm. 7 to the β-reduction relation does not give exactly Prop. 2 but a slight improvement of it, which in the ⟪Beta⟫ case also assumes $x \notin \mathit{FV}\ t_2$. Indeed, the assumption $B \cap (\mathit{Psupp}\ p \cup \mathit{Supp}^{\mathcal{T}} t) = \emptyset$ from Thm. 7 gives in the ⟪Beta⟫ case the assumption

$$G\ \varphi\ B\ (s, s') \iff \begin{array}{ll} (1) & (\exists x, t_1, t_2.\ B = \{x\} \wedge s = Lm\ x\ t_1 \wedge s' = t_1[t_2/x]) \vee \\ (2) & (\exists x, t, t'.\ \varphi\ (t, t') \wedge B = \{x\} \wedge\ s = Lm\ x\ t \wedge s' = Lm\ x\ t') \vee \\ (3) & (\exists t_1, t_2, t_1'.\ \varphi\ (t_1, t_1') \wedge B = \emptyset \wedge s = Ap\ t_1\ t_2 \wedge s' = Ap\ t_1'\ t_2) \vee \\ (4) & (\exists t_1, t_2, t_2'.\ \varphi\ (t_2, t_2') \wedge B = \emptyset \wedge s = Ap\ t_1\ t_2 \wedge s' = Ap\ t_1\ t_2') \end{array}$$

Fig. 2. The operator associated to λ-calculus β-reduction

$x \notin Psupp\ p \cup FV\ (Ap\ (Lm\ x\ t_1)\ t_2)) \cup FV\ (t_1[t_2/x])$, i.e., $x \notin Psupp\ p$ and $x \notin FV\ t_2$. Thus, we obtain as an extra hypothesis in the *induction* proof rule (making induction easier) exactly what Urban et al. must add as an extra hypothesis in the underlying *introduction* rule (making introduction harder).

Note that, for any inductive predicate (regardless of bindings) there is a"tension" between the introduction rules and the induction principle, namely by strengthening or weakening the hypotheses in the defining rules one becomes harder and the other easier to apply. From an abstract standpoint, what a strong induction principle achieves by taking advantage of the binding structure is to have the cake and eat it to, i.e., make induction easier to apply without affecting the introduction rules. This seems connected with the *some/any* principle from Nominal Logic [Gabbay and Pitts 2002, Prop. 3.4] [Pitts 2003, Prop. 4], which states that existential and universal quantification over fresh variables are equivalent (and forms the basis for the *freshness quantifier* [Gabbay and Pitts 2002]). Indeed, pushing this principle through the inductive definition while turning any (implicitly) existentially quantified fresh variables into universally quantified ones, under suitable assumptions about the definition could lead to an alternative proof of strong rule induction.

While the choice of $\mathcal{T}$ is straightforward, the choice of G requires some explanation. First note that, in Fig. 2's definition of G, everything but the treatment of the $B \in \mathcal{P}_{\text{fin}}(Var)$ argument is completely determined by the original definition of the β-reduction predicate $\Rightarrow : LTerm \to LTerm \to Bool$ from Fig. 1. Indeed, if we ignore the treatment of B (the highlighted bits), we obtain the definition of a monotonic operator from $(LTerm^2 \to Bool) \to (LTerm^2 \to Bool)$ that, modulo currying, is exactly the operator underlying the definition of $\Rightarrow$ (as its least fixpoint, via Knaster–Tarski), where the four disjuncts from Fig. 2 correspond to the four rules from Fig. 1.

As for the B argument, its value is also completely determined by virtue of its role: *to store the bound variables that might occur in the conclusions of the rules.* In this case, we have at most one variable, so B will be either a singleton or the empty set. In general, variables may be bound within complex binding structures, e.g., nested record patterns as in the POPLmark Challenge 2B [Aydemir et al. 2005]. We are not interested in the exact form of these structures, but (at least for now) only in the set of variables that they contain.

Remark 9. Above, we argued that B is uniquely determined when we think of it as storing all the bound variables from a rule's conclusion. However, as one of the anonymous reviewers noted, an arbitrary $B \in \mathcal{P}_{\text{fin}}(Var)$ above that minimal value would also work. In other words, we can loosen B upwards, i.e., in the definition of G from Fig. 2 replace the condition $B = \{x\}$ with $x \in B$ in disjuncts (1) and (2) and remove the condition $B = \emptyset$ form disjuncts (3) and (4) (since $\emptyset \subseteq B$ would be vacuous). All the needed checks, including $\mathcal{T}$-refreshability and equivariance, would also succeed for this looser definition of G.

Let us check that these choices of $\mathcal{T}$ and G satisfy the hypotheses of Thm. 7. G is obviously monotonic (as all logical connectives appearing in it are in the positive fragment of first-order logic). And G is equivariant because all operators appearing in it are equivariant.

It remains to check that G is $\mathcal{T}$-refreshable. To this end, let $\varphi : T \to Bool$ be an equivariant predicate, let $B \in \mathcal{P}_{\text{fin}}(Var)$ and $(s, s') \in T = LTerm^2$, and assume $G\ \varphi\ B\ (s, s')$. We must find

$B' \in \mathcal{P}_{\text{fin}}(Var)$ such that $B' \cap Supp^{\mathcal{T}}(s, s') = \emptyset$, i.e., **(i)** $B' \cap (FV\ s \cup FV\ s') = \emptyset$, and **(ii)** $G\ \varphi\ B'\ (s, s')$. We distinguish four cases, depending on which disjunct from G's definition applies to (ii):

(1) Assume $B = \{x\}$, $s = Lm\ x\ t_1$ and $s' = t_1[t_2/x]$ for some x, t_1, t_2. We choose x' to be completely fresh (i.e., fresh for x, t_1 and t_2) and take $B' = \{x'\}$. Now, (i) holds by the choice of x'. Moreover, (ii) holds by virtue of the same disjunct (the first one) in the definition of G holding, with the existential witnesses $x', t_1[x' \leftrightarrow x], t_2$. Indeed:
- $B' = \{x'\}$ holds by definition;
- $s = Lm\ x\ t_1 = Lm\ x'\ (t_1[x' \leftrightarrow x])$ from properties of abstraction;
- $s' = t_1[t_2/x] = t_1[x' \leftrightarrow x][t_2/x']$ from properties of substitution.

(2) Assume $\varphi(t, t')$, $B = \{x\}$, $s = Lm\ x\ t$ and $s' = Lm\ x\ t'$ for some x, t, t'. Like before, we choose x' to be completely fresh and take $B' = \{x'\}$. Again, (i) holds by the choice of x', and (ii) holds by virtue of the same disjunct (the second one) in the definition of G, with the existential witnesses $x', t[x' \leftrightarrow x], t'[x' \leftrightarrow x]$:
- $\varphi\ (t[x' \leftrightarrow x], t'[x' \leftrightarrow x])$ because $\varphi(t, t')$ and φ is equivariant;
- $B' = \{x'\}$ holds by definition;
- $s = Lm\ x\ t = Lm\ x'\ (t[x' \leftrightarrow x])$ from properties of abstraction;
- $s' = Lm\ x\ t' = Lm\ x'\ (t'[x' \leftrightarrow x])$ from properties of abstraction.

(3) Assume $\varphi(t_1, t_1')$, $B = \emptyset$, $s = Ap\ t_1\ t_2$ and $s' = Ap\ t_1'\ t_2$ for some t_1, t_2, t_1'. Then (i) and (ii) hold trivially, taking $B' = \emptyset$ and, in (ii), using the same disjunct (the third one) with t_1, t_2, t_1' as witnesses.

(4) Similar to (3).

6 On Instantiating Our Theorem

As our examples suggest, our criterion is widely applicable. But in addition to the scope question, we are also interested in the formal engineering question on how difficult it is to instantiate this criterion. Fortunately, the instantiation follows well-understood patterns, facilitating automation.

Next, we will extrapolate from our §5 discussion about β-reduction, emphasizing the wider generality of the ideas presented there. The hypothetical scenario we consider is starting with a predicate over syntax with bindings specified inductively via rules, and wishing to deploy our Thm. 7 to obtain a strong rule induction principle for it—which in the case of β-reduction would be Prop. 2.

The operator G associated to our given predicate can be determined from its rules: G is a disjunction consisting of one disjunct for each rule; and each disjunct is an existential, quantifying over all component items (variables, terms, etc.) in the corresponding rule. This process, of extracting from a rule-based specification the underlying operator that ends up capturing the specified predicate as its least fixed point, is well-understood, and has been automated in several theorem provers, including the HOL-based provers HOL4 [HOL 2024; Gordon and Melham 1993], HOL Light [Harrison 2024] and Isabelle/HOL [Nipkow et al. 2002]. In addition, here we need to plug in the value of the set-of-bound-variables argument B, which in each disjunct of G is the set of variables bound (or substituted) in the conclusion of the corresponding rule. This requires knowing the involved binding structures, and can be facilitated by tools that track bindings at datatype-definition time—which include Nominal Isabelle [Urban 2008; Urban and Kaliszyk 2012] (and our own tool we describe in our technical report [van Brügge et al. 2025b, App. G]).

Checking G's monotonicity and equivariance tends to be routine. Most HOL-based provers track monotonicity as part of their inductive definition facilities. Nominal Isabelle tracks equivariance based on its compositionality [Pitts 2013], and Isabelle's Lifting&Transfer tool [Huffman and Kunčar 2013] tracks the related notion of parametricity [Reynolds 1983; Wadler 1989].

The only check that could be non-trivial is that of the $\mathcal{T}$-refreshability of G, which means: We start with an equivariant $\varphi : T \to Bool$, a B and a $t \in T$ such that $G\ \varphi\ B\ t$ holds; and must produce a B' such that $B' \cap Supp^{\mathcal{T}} t = \emptyset$ and $G\ \varphi\ B'\ t$. As hinted in §5, this can proceed via the following heuristic:

Step 1: Because $G\ \varphi\ B\ t$ holds and is expressed as a disjunction of existentials, we obtain some items, usually terms or variables, that satisfy one of the disjuncts, which we will refer to as the "original" items (e.g., x, t, t' in the second disjunct of $G\ \varphi\ B\ t$ in Fig. 2).

Step 2: We pick some completely fresh variables to replace the variables in B, i.e., for each variable x in B we pick a fresh variable x', and take B' to be the corresponding disjoint copy of B (consisting of the "primed" variables). This ensures that $B' \cap Supp^{\mathcal{T}} t = \emptyset$ holds.

Step 3: To prove $G\ \varphi\ B'\ t$, which again is a disjunction of existentials, we prove the same disjunct as the one known to hold for $G\ \varphi\ B\ t$ (e.g., the second disjunct of $G\ \varphi\ B'\ t$ if it is the second disjunct of $G\ \varphi\ B\ t$ that happened to hold), and as witnesses for this disjunct's existentials we plug in the original items (that witnessed the corresponding disjunct of $G\ \varphi\ B\ t$) *in which we swap the original variables x with their fresh counterparts x' as appropriate*. Here, "as appropriate" means that swapping only takes place if the variable x is either equal to the considered original item, or that item is in the scope of its binding. Thus, for example, for Fig. 2's first disjunct we replace x with x' and t_1 with $t_1[x \leftrightarrow x']$, but t_2 stays as it is (because the latter is not in the scope of the bound variable x). It remains to verify the disjunct of $G\ \varphi\ B'\ t$ whose existentials have been instantiated with these witnesses. For example, in the case of Fig. 2's second disjunct, knowing that $\varphi(t, t')$, $B = \{x\}$, $s = Lm\ x\ t$ and $s = Lm\ x'\ t'$ hold, we want to verify that $\varphi(t[x \leftrightarrow x'], t'[x \leftrightarrow x'])$, $B' = \{x'\}$, $s = Lm\ x'\ (t[x \leftrightarrow x'])$ and $s = Lm\ x'\ (t'[x \leftrightarrow x'])$ also hold. Among these goals to be proved:

- those involving B' follow from the prior knowledge about B and the construction of B' (e.g., $B' = \{x'\}$ follows from $B = \{x\}$);
- those involving the occurrences of the predicate, e.g., $\varphi(t[x \leftrightarrow x'], t'[x \leftrightarrow x'])$, follow from the corresponding fact in the original disjunct, e.g., $\varphi(t, t')$, and the assumed equivariance of φ.

As for the other goals, such as $s = Lm\ x\ t$ implying $s = Lm\ x'\ (t[x \leftrightarrow x'])$, which amounts to $Lm\ x\ t = Lm\ x'\ (t[x \leftrightarrow x'])$, they say that the original items can be replaced in certain contexts by the "refreshed" (swapped) items. For these, we have no general recipe but an empirical observation validated on many examples: These goals tend to be reducible to standard properties of the syntactic operators (constructors, swapping, substitution, etc.).

Remark 10. A crucial part of the above heuristic for checking $\mathcal{T}$-refreshability is assuming that the predicate argument φ of G is equivariant and holds for some "original" items, and wanting to prove that it holds for modifications of these items where the variables from B are swapped "as appropriate", i.e., swapped or not depending on their being in the scope of bindings in the rules' conclusions. (Note that φ intuitively stands for the inductively defined predicate during iteration through G.) Favorable situations that work out of the box are when, in the hypotheses of each defining rule, each occurrence of the inductively defined predicate is: **(A)** either applied to items that are *all not in* the scope of bound variables in the conclusion, yielding trivial goals such as "$\varphi(t, t')$ implies $\varphi(t, t')$", or **(B)** applied to items that are *all in* the scope of bound variables in the conclusion, yielding goals such as "$\varphi(t, t')$ implies $\varphi(t[x \leftrightarrow x'], t'[x \leftrightarrow x'])$" which follow from φ's equivariance. Otherwise, we encounter a hybrid situation, i.e., an in-hypotheses occurrence of the inductively defined predicate that is **(C)** applied to some items in, and to some items not in the scope of bound variables in the conclusion. Then, we end up with hybrid goals such as "$\varphi(t, t')$ implies $\varphi(t[x \leftrightarrow x'], t')$". In these cases, the only way forward is if the given rule guarantees, perhaps via a side-condition, the freshness of the original variables for the offending original terms (i.e., those subjected to swapping), e.g., the freshness of x for t—because, x' being fresh as well, we would have $t[x \leftrightarrow x'] = t$ so we could fall back on case (A).

Remark 11. Let us see what problems we would incur with our β-reduction example if we tried to check that G satisfies not $\mathcal{T}$-refreshability but the stronger condition that we called $\mathcal{T}$-freshness (in Def. 6). The latter requires that $G\ \varphi\ B\ t$ implies $B \cap Supp^{\mathcal{T}} t = \emptyset$, i.e., that $B \cap Supp^{\mathcal{T}} t = \emptyset$ follows from each disjunct in the definition of $G\ \varphi\ B\ t$, corresponding to a rule in the inductive definition of β-reduction. So we want all the variables in B, i.e., those appearing bound in the rule's conclusion, to be prevented from (also) appearing free in the rule's conclusion. This works for all the rules except the first one in Fig. 1 (corresponding to the first disjunct in Fig. 2), where x which appears bound in the conclusion is not prevented from also appearing free in the conclusion, e.g., within t_2. Thus, a fix to get $\mathcal{T}$-freshness would be adding the side-condition that x be fresh for t_2, as seen in the rule (Beta') from §2; and a similar situation occurs with the "Parallel Beta" rule (ParBeta) from §2, which to validate $\mathcal{T}$-freshness must become (ParBeta'). In fact, as detailed in our technical report [van Brügge et al. 2025b, App. A], our $\mathcal{T}$-freshness generalizes Urban et al. [2007]'s criterion. The next section further explores the difference between the two criteria.

7 More on the Comparison with the Urban et al. Criterion

As stated at the end §2, our Thm. 7 improves on Urban et al. [2007]'s result in two ways: (1) not necessitating the addition of side-conditions to capture concrete systems and (2) going beyond syntactic format for the rules. In this section, taking advantage of the availability of more concepts and notation, we will further illustrate what improvement (1) amounts to by going into finer details.

We consider again the standard β-reduction relation described in Fig. 1. So our theorem applies to this relations' definition as is, whereas in order to apply Urban et al.'s criterion (Theorem 1 from [Urban et al. 2007]) one requires a modification of the definition, namely the addition of the side-condition $x \notin FV\ t_2$ to the (Beta) rule, i.e., the replacement of (Beta) with the rule (Beta') shown below:

$$\frac{}{Ap\ (Lm\ x\ t_1)\ t_2 \Rightarrow t_1[t_2/x]}\ \frac{\text{(Beta')}}{[x \notin FV\ t_2]}$$

It turns out that the modified system can be proved equivalent with (equal to) the original one—and Urban et al. noted that this tends to be the case in concrete examples, but they left open the problem of proving that in a general setting (such as the setting, based on a format for schematic rules).

Let us see how to prove that the above two concrete systems, the original one and the one modified by having (Beta') replacing (Beta), are equivalent. Clearly the original one is at least as strong as the modified one. Conversely, to prove that the modified one is at least as strong as the original one, we essentially need to prove that (Beta) can be "simulated" by (Beta'). And indeed, this intuitively seems to be the case because the bound variable x in (Beta) can in principle be renamed to something fresh for t_2, and this renaming should be immaterial (since terms are quotiented to α-equivalence). However, we cannot simply invoke such a renaming without further argumentation. This is because, in (Beta) and (Beta'), the term t_1 appears not only inside the scope of a λ-bound x (within $Lm\ x\ t_1$) by also outside this scope (within $t_1[2_2/x]$)—so while the first occurrence of t_1 has x bound, the second occurrence "exposes" the name x. We must take this into account when doing the inference of (Beta) from (Beta'), which goes as follows: We assume (Beta') holds. To prove (Beta), let x be a variable and t_1, t_2 be terms; we need to show $Ap\ (Lm\ x\ t_1)\ t_2 \Rightarrow t_1[t_2/x]$. To this end, we pick a completely fresh variable, say x', and define t_1' by swapping (or alternatively substituting) x with x' in t_1, namely $t_1' = t_1[x \leftrightarrow x']$. Then using the properties of swapping and substitution and the fact that x' is fresh for t_1, we obtain that $t_1'[t_2/x'] = t_1[t_2/x]$; and using the properties of λ-abstraction (stemming from α-equivalence), we obtain that $Lm\ x'\ t_1' = Lm\ x\ t_1$. This allows us to infer the desired instance of (Beta), namely $(Lm\ x\ t_1)\ t_2 \Rightarrow t_1[t_2/x]$, from an instance of (Beta'), namely $(Lm\ x'\ t_1')\ t_2 \Rightarrow t_1'[t_2/x']$; the latter is indeed an instance of (Beta') since x' is fresh for t_2.

$$t \Longrightarrow t \text{ (Refl)} \qquad \frac{t \Longrightarrow t'}{Lm\ x\ t \Longrightarrow Lm\ x\ t'} \text{ (Xi)}$$

$$\frac{t_1 \Longrightarrow t_1' \qquad t_2 \Longrightarrow t_2'}{Ap\ t_1\ t_2 \Longrightarrow Ap\ t_1'\ t_2'} \text{ (Ap)} \qquad \frac{t_1 \Longrightarrow t_1' \qquad t_2 \Longrightarrow t_2'}{Ap\ (Lm\ x\ t_1)\ t_2 \Longrightarrow t_1'[t_2'/x]} \text{ (ParBeta)}$$

Fig. 3. λ-calculus parallel β-reduction

In the case of (Beta) versus (Beta'), the technicalities were relatively simple thanks to dealing with an axiom (i.e., a rule with no hypotheses), but when dealing with proper rules (as is more often the case) inference is more difficult. We illustrate this with the parallel β-reduction relation briefly mentioned in §2, which was Urban et al.'s initial motivating example. Its definition is shown in Fig. 3. Again, our theorem applies to this definition as is, whereas Urban et al.'s theorem requires the addition of the side-condition $x \notin FV\ t_2 \cup FV\ t_2']$ to the (ParBeta) rule, i.e., the replacement of (ParBeta) with the rule (ParBeta') shown below:

$$\frac{t_1 \Longrightarrow t_1' \qquad t_2 \Longrightarrow t_2'}{Ap\ (Lm\ x\ t_1)\ t_2 \Longrightarrow t_1'[t_2'/x]}\ [x \notin FV\ t_2 \cup FV\ t_2'] \text{ (ParBeta')}$$

Now, it is not even true that, in isolation (that is, regardless of what the other rules of the system are) the rule (ParBeta) is inferable from the rule (ParBeta'). What *is* inferable from (ParBeta'), applying an argument similar to the one sketched above for (Beta) versus (Beta') (that is, picking a fresh x' and using properties of substitution, swapping and constructors), is only a modification of (ParBeta) that replaces the hypotheses $t_1 \Longrightarrow t_1'$ and $t_2 \Longrightarrow t_2'$ with $t_1[x \leftrightarrow x'] \Longrightarrow t_1'[x \leftrightarrow x']$ and $t_2[x \leftrightarrow x'] \Longrightarrow t_2'[x \leftrightarrow x']$ for some fresh x'. Then, after we prove equivariance for the entire system featuring (ParBeta') and the other rules (so depending on the well-behavedness of the other rules as well), we can replace the modified hypotheses with the original hypotheses of (ParBeta)—concluding the proof that the two versions are equivalent (since the opposite direction, i.e., moving from (ParBeta) to (ParBeta'), is again trivial).

Note that the above arguments for getting rid of certain side-conditions involved an equivariance proof, and also some specific properties of the operators participating in the rules, such as substitution. Our strong rule induction criterion, Thm. 7, can be regarded as providing a generalization of such arguments baked into the argument for the soundness of strong rule induction.

A final note about the above rule (ParBeta'): The $x \notin FV\ t_2'$ part of the added side-condition is seen to be redundant also because parallel β-reduction can be proved to not any new free variables (when moving from left to right), so $x \notin FV\ t_2'$ follows from $x \notin FV\ t_2$. But general-purpose criteria such as Urban et al.'s and ours are not addressing such specific semantic properties though (nor do they assume, of course, that the defined predicate takes the form of a transition relation). In particular, while our criterion does not require the addition of side-conditions, it does not provide a mechanism for detecting redundant side-conditions when already part of the original rules.

Overview of the Next Two Sections. In what follows, we validate, challenge and refine the meta-theory through examples that exhibit more complexity than the λ-calculus along several directions: scope extrusion and complex side-conditions (π-calculus, §8.1), environments (System $F_{<:}$, §8.2), and terms with infinitely many variables (infinitary FOL §9.1 and λ-calculus §9.3). While the π-calculus example showcases the improvements of our criterion over the state of the art, we chose to present the other examples because they have challenged this criterion, inspiring further improvements and generalizations: making inductive information available for refreshability (§8.3), allowing infinitary structures (§9.2), and considering binders explicitly while loosening equivariance (§9.4).

8 More Examples and Refinements

For the syntaxes in this section, we will implicitly use standard notions such as permutation and free variables. They all form nominal sets similarly to how the λ-calculus syntax does.

8.1 Example: π-Calculus

In this subsection, variables will sometimes be called "names" or "channels". We also let a, b (in addition to x, y, z) range over variables. The set *Proc*, of π-calculus *processes* [Milner 1999; Milner et al. 1992], ranged over by P, Q, R etc., is described by a grammar of the following form (where we omit the constructors that will play no role in our discussion):

$$P \quad ::= \quad \ldots \mid P \parallel Q \mid !P \mid \overline{a}\,x.\,P \mid a(x).\,P \mid \nu(x).\,P$$

We assume that x is bound in P within processes of the form $a(x).\,P$ and $\nu(x).\,P$; and processes are equated modulo the induced alpha-equivalence. The shown constructors are, in order: parallel composition, replication, output (of name x on the channel a), input (of a generic name x on channel a), and restriction/hiding (of the name x).

The set *Act* of *actions*, ranged over by α, is given by the grammar:

$$\alpha \ ::= \ \tau \mid a\,x \mid \overline{a}\,x \mid a(x) \mid \overline{a}(x)$$

The above are, in order: the silent action, the input of a (free) name x on channel a, the output of a (free) name x on channel a, the symbolic input of a (bound) name x on channel a, and the output of a bound name x on channel a. The first three types will be called *free actions*; we let $fra\ \alpha$ express the fact that α is a free action.

We let $ns\ \alpha$, the set of names of an action α, consist of all the names appearing in that action (so $ns\ \alpha$ is empty if $\alpha = \tau$ and otherwise it has at most two elements). We also let $bns\ \alpha$, the set of *bound names of* α, be $\{x\}$ if α has the form $a(x)$ or $\overline{a}(x)$, and $\emptyset$ otherwise. And $fns\ \alpha$, the set of *free names of* α, be $\{a\}$ if α has the form $a(x)$ or $\overline{a}(x)$, and $ns\ \alpha$ otherwise. In particular, we have $ns\ \alpha = bns\ \alpha \cup fns\ \alpha$, though $bns\ \alpha$ and $fns\ \alpha$ may not be disjoint.

A process can take an action by consuming one of its communicating prefixes ($\overline{a}\,x.$ or $a(x).$) and transitioning to a remainder process. This is described by an inductively defined transition relation, using rules including ones shown in Fig. 4.

There are two main variants of operational semantics for the π-calculus—early-instantiation and late-instantiation—depending on whether input instantiation is exhibited "early" for single processes or "late" during communication. Fig. 4 shows the binding-interesting rules for both variants. (For conciseness, we used a notion of action that is broad enough to accommodate both variants.)

A binding behavior characteristic to the π-calculus is *scope extrusion*: Via the rule (Open), a process, say $\nu(x).\,Q$, "opens" the scope of a previously bound variable x; then, via the rule (CloseLeftE) or (CloseLeftL) (or their symmetrics), the scope is "closed" after another process P receives this bound name. At the end of this scope opening and closing session, a name x that was previously known to the process Q alone has now been shared with P—becoming a shared secret between the remainder processes P' and Q'.

Remark 12. In a naive formalization of the transition relation, namely as a *ternary* relation, a rule such as (Open) is known to be problematic for formal reasoning, essentially because it is resistant to strong induction [Bengtson 2010]. In fact, we can explain this problem in terms of our §6 heuristic for proving $\mathcal{T}$-refreshability. We would get stuck along the lines sketched in Remark 10: attempting to prove, for an equivariant predicate $\varphi : Proc \times Act \times Proc \to Bool$ and a fresh x', the hopeless goal "$\varphi\,(P, \overline{a}\,x, P')$ implies $\varphi\,(P[x \leftrightarrow x'], \overline{a}\,x', P')$" while knowing that $a \neq x$ but *not* that x is fresh for P. (And while the "fix" of adding to (Open) the side-condition that x be fresh for P would indeed enable strong induction, it would also destroy the intended semantics by preventing P from sending any of

$$a(x).\,P \xrightarrow{a\,y} P[y/x] \text{ (InpE)} \qquad \frac{P \xrightarrow{a\,x} P' \quad Q \xrightarrow{\overline{a}\,x} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \text{ (ComLeftE)} \qquad \frac{P \xrightarrow{a\,x} P' \quad Q \xrightarrow{\overline{a}(x)} Q'}{P \parallel Q \xrightarrow{\tau} \nu(x).\,(P' \parallel Q')} \text{ (CloseLeftE)}\ [x \notin \{a\} \cup FV\ P]$$

Rules specific to the early-instantiation semantics

$$a(x).\,P \xrightarrow{a(x)} P \text{ (InpL)} \qquad \frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\overline{a}\,y} Q'}{P \parallel Q \xrightarrow{\tau} P'[y/x] \parallel Q'} \text{ (ComLeftL)} \qquad \frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\overline{a}(x)} Q'}{P \parallel Q \xrightarrow{\tau} \nu(x).\,(P' \parallel Q')} \text{ (CloseLeftL)}$$

Rules specific to the late-instantiation semantics

$$\frac{P \xrightarrow{\overline{a}\,x} P'}{\nu(x).\,P \xrightarrow{\overline{a}(x)} P'} \text{ (Open)}\ [a \neq x] \qquad \frac{P \xrightarrow{\alpha} P'}{\nu(x).\,P \xrightarrow{\alpha} \nu(x).\,P'} \text{ (ScopeFree)}\ [fra\ \alpha,\ x \notin ns\ \alpha]$$

$$\frac{P \xrightarrow{\overline{a}(x)} P'}{\nu(y).\,P \xrightarrow{\overline{a}(x)} \nu(y).\,P'} \text{ (ScopeBound)}\ [y \notin \{a, x\},\ x \notin FV\ P \cup \{a\}] \qquad \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \text{ (ParLeft)}\ [bns\ \alpha \cap FV(P, Q) = \emptyset]$$

Rules common to both styles of semantics

Fig. 4. π-calculus transition relation

$$a(x).\,P \xrightarrow{a\,y} P[y/x] \text{ (InpE')}\ [x \notin \{a, y\}] \qquad \frac{P \xrightarrow{a\,x} P' \qquad Q \xrightarrow{\overline{a}(x)} Q'}{P \parallel Q \xrightarrow{\tau} \nu(x).(P' \parallel Q')} \text{ (CloseLeftE')}\ [x \notin \{a\} \cup FV(P,Q)$$

$$\frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\overline{a}\,y} Q'}{P \parallel Q \xrightarrow{\tau} P'[y/x] \parallel Q'} \text{ (ComLeftL')}\ [x \notin FV(P, Q, Q')]$$

$$\frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\overline{a}(x)} Q'}{P \parallel Q \xrightarrow{\tau} \nu(x).\,(P' \parallel Q')} \text{ (CloseLeftL')}\ [x \notin FV(P, Q)] \qquad \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \text{ (ParLeft')}\ [bns\ \alpha \cap FV(P, Q) = \emptyset, bns\ \alpha \cap fns\ \alpha = \emptyset]$$

Fig. 5. π-calculus transitions augmented to accommodate prior state-of-the-art strong rule induction

its known (i.e., free) names.) This is not a problem with our criterion, but a situation where applying Barendregt's convention would be unsound; a similar example is given Urban et al. [2007, p.38].

An elegant solution to the above problem comes from noting the following about the intended semantics: that any name which is bound in the action labeling the transition, e.g., a name x sent via an $\overline{a}(x)$ action, has its identity "hidden"; in particular, until further extruding actions, is unavailable to any other process besides the one that sends it and the one that receives it. This is best modeled syntactically by assuming that such a name x *gets bound from within the action into the remainder process*. Thus, in the conclusion $\nu(x).\,P \xrightarrow{\overline{a}(x)} P'$ of (Open), we think of the occurrence of x in $\overline{a}(x)$ as binding any (free) occurrence of x in P'. This solution was pursued in his thesis by Bengtson [Bengtson 2010], who (crediting Milner et al. for the idea [Milner 1993; Milner et al. 1992]) formalizes the π-calculus transition relation not as a ternary relation between a source process, an action and a target process, but as a binary relation between a source process and a commitment, the latter being a pair (action, remainder process) up to alpha-equivalence.

Following Bengtson, we thus define the set *Com* of *commitments* to consist of pairs $C = (\alpha, P)$ up to alpha-equivalence. That is, commitments are generated by the (nonrecursive) grammar

$$C ::= (\tau, P) \mid (a\,x, P) \mid (\overline{a}\,x, P) \mid (a(x), P) \mid (\overline{a}(x), P)$$

(having one production for each action type) with the assumption that, in a commitment of the form $(a(x), P)$ or $(\overline{a}(x), P)$, x is bound in P; and commitments are identified modulo alpha-equivalence.

Under this commitment-based view, Fig. 4 stays the same, but now we read $P \xrightarrow{\alpha} P'$ as notation for $tran\ P\ (\alpha, P')$, where $tran : Proc \to Com \to Bool$ is a (binary) relation between *Proc* and *Com*. Thus, the rules in Fig. 4 give an inductive definition of *tran*. In this setting, the problem with (Open) from Remark 12 vanishes, because now both occurrences of x from its conclusion are bound: one binding in P and one in P'. Hence our heuristic for $\mathcal{T}$-refreshability succeeds, as it just needs to check "$\varphi\ P\ (\overline{a}\,x)\ P'$ implies $\varphi\ (P[x \leftrightarrow x'])\ (\overline{a}\,x')\ (P'[x \leftrightarrow x'])$", an instance of φ's equivariance.

And indeed, applying Thm. 7 to the inductive rules from Fig. 4, we obtain the desired strong rule induction, where in the inductive hypotheses we can assume that all the bound variables referenced in these rules are fresh for the parameters. For example, here is the strong rule induction we obtain if we consider that the transition relation is defined by a particular selection of the Fig. 4 rules, namely (InpE), (CloseLeftE) and (ComLeftL), (CloseLeftL) and (ParLeft):

Prop 13. Let $(P, Psupp : P \to \mathcal{P}_{\text{fin}}(Var))$ be a parameter structure. Let $\varphi : P \to Proc \to Com \to Bool$ and assume the following hold:

- (InpE): $\forall p, a, x, y, P, Q.\ x \notin Psupp\ p \wedge x \notin \{a, y\} \longrightarrow \varphi\ p\ (a(x).P)\ (\overline{a}\,y, P[y/x])$
- (CloseLeftE): $\forall p, a, x, P, P', Q, Q'.\ x \notin Psupp\ p \wedge x \notin FV\ Q \wedge x \neq a \wedge x \notin FV\ P \wedge$
  $(P \xrightarrow{a\,x} P') \wedge (\forall q.\ \varphi\ q\ P\ (a\,x, P')) \wedge (Q \xrightarrow{\overline{a}(x)} Q') \wedge (\forall q.\ \varphi\ q\ Q\ (\overline{a}(x), Q')) \longrightarrow$
  $\varphi\ p\ (P \parallel Q)\ (\tau, P' \parallel Q')$
- (ComLeftL): $\forall p, a, x, y, P, P', Q, Q'.\ x \notin Psupp\ p \wedge x \notin FV\ (P, Q, Q') \wedge$
  $(P \xrightarrow{a(x)} P') \wedge (\forall q.\ \varphi\ q\ P\ (a(x), P')) \wedge (Q \xrightarrow{\overline{a}\,y} Q') \wedge (\forall q.\ \varphi\ q\ Q\ (\overline{a}\,y, Q')) \longrightarrow$
  $\varphi\ p\ (P \parallel Q)\ (\tau, P'[y/x] \parallel Q')$
- (CloseLeftL): $\forall p, a, x, P, P', Q, Q'.\ x \notin Psupp\ p \wedge x \notin FV\ (P, Q) \wedge$
  $(P \xrightarrow{a(x)} P') \wedge (\forall q.\ \varphi\ q\ P\ (a(x), P')) \wedge (Q \xrightarrow{\overline{a}(x)} Q') \wedge (\forall q.\ \varphi\ q\ Q\ (\overline{a}(x), Q')) \longrightarrow$
  $\varphi\ p\ (P \parallel Q)\ (\tau, P' \parallel Q')$
- (ParLeft): $\forall p, \alpha, P, P', Q.\ bns\ \alpha \cap Psupp\ p = \emptyset \wedge bns\ \alpha \cap fns\ \alpha = \emptyset \wedge bns\ \alpha \cap FV(P, Q) = \emptyset \wedge$
  $(P \xrightarrow{\alpha} P') \wedge (\forall q.\ \varphi\ q\ P\ (\alpha, P')) \longrightarrow \varphi\ p\ (P \parallel Q)\ (\alpha, P' \parallel Q)$

Then $\forall p, P, \alpha, P'.\ (P \xrightarrow{\alpha} P') \longrightarrow \varphi\ p\ P\ (\alpha, P')$.

On the other hand, using the state of the art [Urban et al. 2007] as implemented in Nominal Isabelle (which Bengtson used in his formalization [Bengtson 2012]), to get the same result one needs to augment the rules with side-conditions as highlighted in Fig. 5. These would ensure that the system satisfies not only $\mathcal{T}$-refreshability, but also $\mathcal{T}$-freshness. Indeed, as we discussed in Remark 11, $\mathcal{T}$-freshness in concrete examples amounts to the variables appearing bound in the conclusion of a rule being prevented from also appearing free in that conclusion. For example, $\mathcal{T}$-freshness does not hold for the rule (CloseLeftL) from Fig. 4 because x, which appears bound in the conclusion, can also appear free there, namely within P and Q—so to make $\mathcal{T}$-freshness hold one must add the side-condition highlighted in (CloseLeftL') from Fig. 5. Unlike in the situation from Remark 12, and like in those from Remark 11, here these fixes (required for $\mathcal{T}$-freshness but not for $\mathcal{T}$-refreshability) do not destroy the intended meaning of the definitions, but introduce unnecessary clutter.

Some versions of π-calculus [Sangiorgi and Walker 2001] distinguish between structural and operational rules—they too admit strong rule induction (as we illustrate on an example in our technical report [van Brügge et al. 2025b, App. B]).

$$\frac{wf\ \Gamma\ \ FV\ S \subseteq dom\ \Gamma}{\Gamma \vdash S <: Top}\ (\text{Top}) \qquad \frac{wf\ \Gamma \quad X \in dom\ \Gamma}{\Gamma \vdash (TVr\ X) <: (TVr\ X)}\ (\text{Refl-TV}) \qquad \frac{X <: S \in \Gamma \quad \Gamma \vdash S <: T}{\Gamma \vdash S <: T}\ (\text{Trans-TV})$$

$$\frac{\Gamma \vdash T_1 <: S_1 \qquad \Gamma \vdash S_2 <: T_2}{\Gamma \vdash (S_1 \to S_2)\ <:\ (T_1 \to T_2)}\ (\text{Arrow}) \qquad \frac{\Gamma \vdash T_1 <: S_1 \qquad \Gamma, X <: T_1 \vdash S_2 <: T_2}{\Gamma \vdash (\forall X <: S_1.\ S_2)\ <:\ (\forall X <: T_1.\ T_2)}\ (\text{All})$$

Fig. 6. System $\mathsf{F}_{<:}$ subtyping

$$\begin{array}{l} G\ \varphi\ B\ (\Gamma, S', T') \iff \\ (\exists S.\ B = \emptyset \wedge S' = S \wedge T' = Top) \vee (\exists X.\ B = \emptyset \wedge S' = TVr\ X \wedge T' = TVr\ X) \vee \\ (\exists X, Y, T.\ X <: Y \in \Gamma \wedge \varphi\ (\Gamma, TVr\ Y, T) \wedge B = \emptyset \wedge S' = TVr\ X \wedge T' = T) \vee \\ (\exists S_1, S_2, T_1, T_2.\ \varphi\ (\Gamma, T_1, S_1) \wedge \varphi\ (\Gamma, T_2, S_2) \wedge B = \emptyset \wedge S' = (S_1 \to S_2) \wedge T' = (T_1 \to T_2)) \vee \\ (\exists X, S_1, S_2, T_1, T_2.\ \varphi\ (\Gamma, T_1, S_1) \wedge \varphi\ ((\Gamma, X <: T_1), S_2, T_2) \wedge B = \{X\} \wedge S' = (\forall X <: S_1.\ S_2) \wedge T' = (\forall X <: T_1.\ T_2)) \end{array}$$

Fig. 7. The operator associated to System $\mathsf{F}_{<:}$ subtyping

8.2 Example: System $\mathsf{F}_{<:}$ Subtyping

Next we look at the subtyping relation for System $\mathsf{F}_{<:}$ [Aydemir et al. 2005; Cardelli et al. 1994; Curien and Ghelli 1992], an example combining type bindings with environment bindings.

In this subsection, the variables in *Var* will stand for type variables, and X, Y, Z etc. will range over them. The set *Type* of *types*, ranged over by S, T etc., is generated by the following grammar:

$$T\ ::=\ TVr\ X\ \mid\ Top\ \mid\ T \to S\ \mid\ \forall X <: T.\ S$$

So a type is either a (type) variable, or the maximum type *Top*, or a function type, or a universal type. We assume that, in a universal type $\forall X <: T.\ S$, the variable X is bound in S (but not in T); and types are equated modulo the induced notion of alpha-equivalence.

A *(typing) environment* Γ is a list of pairs variable-type, (X, T), written $X <: T$. *Env* denotes the set of environments. The domain *dom* Γ of an environment consists of all the variables X for which some $X <: T$ is in Γ. An environment is said to be *well-formed*, written *wf* Γ, if whenever Γ has the form $\Gamma', X <: T, \Gamma''$, we have that $X \notin dom\ \Gamma'$ and $FV\ T \subseteq dom\ \Gamma'$—i.e., thinking of the environment as growing left-to-right with pairs, any new pair $X <: T$ must be such that X is fresh and T does not bring new (free) variables. *Subtyping* is a ternary relation between environments, types and types, written $\Gamma \vdash S <: T$, defined inductively in Fig. 6. On the way to instantiating Thm. 7 to this system, we obtain the operator G shown in Fig. 7. Thm. 7's conclusion would give us the following induction principle, avoiding parameter variables in the ⟪All⟫ case:

Prop 14. Let $(P, Psupp : P \to \mathcal{P}_{\mathsf{fin}}(Var))$ be a parameter structure. Let $\varphi : P \to Env \to Type \to Type \to Bool$ and assume that:

- [cases different from ⟪All⟫ omitted, as they don't involve binders]
- ⟪All⟫: $\forall p, X, S_1, S_2, T_1, T_2.\ X \notin Psupp\ p \wedge X \notin FV(\Gamma, S_1, T_1) \wedge$
  $\Gamma \vdash T_1 <: S_1 \wedge (\forall q.\ \varphi\ q\ \Gamma\ T_1\ S_1) \wedge \Gamma, X <: T_1 \vdash S_2 <: T_2 \wedge (\forall q.\ \varphi\ q\ (\Gamma, X <: T_1)\ S_2\ T_2) \longrightarrow$
  $\varphi\ p\ \Gamma\ (\forall X <: S_1.\ S_2)\ (\forall X <: T_1.\ T_2)$

  Then $\forall p, \Gamma, S, T.\ \Gamma \vdash S <: T \longrightarrow \varphi\ p\ \Gamma\ S\ T$.

However, when attempting to check Thm. 7's hypotheses, we get stuck at $\mathcal{T}$-refreshability. Namely, when deploying the heuristic sketched in §6, we encounter a problem with Fig. 6's (All) rule, i.e., with the fifth disjunct in Fig. 7's definition of G. While focusing on the second hypothesis of the (All) rule, for an equivariant $\varphi : Env \times Type \times Type \to Bool$, we know that **(i)** X' is fresh and **(ii)** $\varphi\ ((\Gamma, X <: T_1),\ S_2,\ T_2)$, and want to prove **(iii)** $\varphi\ ((\Gamma, X' <: T_1),\ S_2[X \leftrightarrow X'],\ T_2[X \leftrightarrow X'])$. (Note that, in (iii), Γ and T_1 are not subject to swapping, because in (All)'s conclusion they are not in the scope of X's binding.) However, φ's equivariance and (ii) only ensure

(iii') $\varphi\ ((\Gamma[X \leftrightarrow X'], X' <: T_1[X \leftrightarrow X']), S_2[X \leftrightarrow X'], T_2[X \leftrightarrow X'])$,

i.e., the variation of (iii) where swapping *is* applied to Γ and T_1. In short, we fall under one of

those "hybrid" situations (case (C)) discussed in Remark 10. Since X' is fresh, we would have $\Gamma[X \leftrightarrow X'] = \Gamma$ and $T_1[X \leftrightarrow X'] = T_1$, hence (iii) would follow from (iii') and the problem would be solved, but only provided that: **(iv)** X is also fresh for Γ and T_1.

But currently there is no way to prove (iv)—which is a shame, because we *can* prove that $\Gamma, X <: T_1 \vdash S_2 <: T_2$ implies (iv), namely as follows: First, we prove by standard induction that, for all Γ', S, T, $\Gamma' \vdash S <: T$ implies $wf\ \Gamma'$. So $\Gamma, X <: T_1 \vdash S_2 <: T_2$ implies $wf\ (\Gamma, X <: T_1)$, which by the definition of wf implies that X is fresh for Γ and $FV\ T \subseteq FV\ \Gamma$, ultimately implying that X is fresh for T as well.

8.3 An Inductively Strengthened Criterion

Thus, we would solve the problem if we could take advantage of properties of the inductively defined predicate when checking the conclusion of the $\mathcal{T}$-refreshability condition. Stepping back into §4's general setting, we are led to a weakening of $\mathcal{T}$-refreshability (highlighting the difference from our original definition, part of Def. 6):

Def 15. Given a nominal set $\mathcal{T} = (T, \_[]^{\mathcal{T}})$ and an operator $G : (T \to Bool) \to (\mathcal{P}_{\text{fin}}(Var) \to T \to Bool)$, we say that G is *weakly $\mathcal{T}$-refreshable* when, for all $\varphi : T \to Bool$ such that $\forall t \in T.\ \varphi\ t \longrightarrow I_G\ t$, for all $B \in \mathcal{P}_{\text{fin}}(Var)$ and $t \in T$, if φ is equivariant and $G\ \varphi\ B\ t$ then there exists $B' \in \mathcal{P}_{\text{fin}}(Var)$ such that $B' \cap Supp^{\mathcal{T}}\ t = \emptyset$ and $G\ \varphi\ B'\ t$.

Since in the statement of $\mathcal{T}$-refreshability, φ morally stands for the inductively defined predicate I_G, adding the hypothesis that φ actually implies I_G makes sense. And indeed, with a bit of proof mining we can strengthen Thm. 7 to use this weaker notion:

Thm 7 strengthened. Let $\mathcal{T} = (T, \_[]^{\mathcal{T}})$ be a nominal set and $G : (T \to Bool) \to (\mathcal{P}_{\text{fin}}(Var) \to T \to Bool)$ be monotonic, $\mathcal{T}$-equivariant and weakly $\mathcal{T}$-refreshable. Then Thm. 7's conclusion holds.

Proof. The only place in the proof of Thm. 7 where we use $\mathcal{T}$-refreshability is when proving **(3)** $\forall t.\ I_G\ t \longrightarrow I'_G\ t$, at a time when we have already proved the converse **(1)** $\forall t.\ I'_G\ t \longrightarrow I_G\ t$, and have also proved that **(2)** I'_G is equivariant. As part of the inductive proof of (3), fixing B and t and assuming **(iii)** $G\ I'_G\ B\ t$, we applied $\mathcal{T}$-refreshability to (2) and (iii) to obtain B' such that $B' \cap Supp^{\mathcal{T}}\ t = \emptyset$ and $G\ \varphi\ B'\ t$. But we can instead apply weak $\mathcal{T}$-refreshability to (2), (iii) and (1) to the same effect. □

In conclusion, the strengthened version of Thm. 7 assumes weak $\mathcal{T}$-refreshability instead of $\mathcal{T}$-refreshability, which allows one to take advantage of inductive information when instantiating the theorem. And indeed, the System $F_{<:}$ typing example is now covered, in that Prop. 14 is a consequence of the strengthened Thm. 7: Going back to the discussion at the end of §8.2, there the extra hypothesis $\forall t \in T.\ \varphi\ t \longrightarrow I_G\ t$ means that (ii) implies $\Gamma, X <: T_1 \vdash S_2 <: T_2$, which fills the pointed gap.

9 Strong Rule Induction for Infinitary Structures with Bindings

While our strong induction criterion discussed so far covers the vast majority of the cases of interest, it is restricted to *finitary* structures modeled as nominal sets. However, infinitary structures featuring bindings have also been studied, and they too are subjected to inductive definitions and proofs that must cope with these bindings. Examples include infinitary extensions of first-order logic (FOL) [Dickmann 1985; Keisler 1971; Marker 2016] (§9.1), a standard variant of Milner's Calculus of Communicating Systems (CCS) [Milner 1989] featuring infinitary choice (sum) and bindings of input variables, versions of Hennessy-Milner logic featuring infinitary conjunctions and bindings for recursion and/or quantification [Hennessy and Stirling 1985], and infinitary higher-order rewriting and proof theory [Joachimski 2001]. Considering such infinitary logics and systems will lead to an extension of our result that employs an an infinitary variation of nominal sets (§9.2). Finally, we will

$$\frac{f \in \Delta}{\Delta \vdash f}\ (\text{Hyp}) \qquad \frac{\forall f \in F.\ \Delta \vdash f}{\Delta \vdash \mathit{Conj}\ F}\ (\text{Conj-I}) \qquad \frac{\Delta \vdash \mathit{Conj}\ F \quad f \in F}{\Delta \vdash f}\ (\text{Conj-E}) \qquad \frac{\Delta, f \vdash \bot}{\Delta \vdash \mathit{Neg}\ f}\ (\text{Neg-I})$$

$$\frac{\Delta \vdash \mathit{Neg}\ f \quad \Delta \vdash f}{\Delta \vdash \bot}\ (\text{Neg-E}) \qquad \frac{\Delta \vdash f}{\Delta \vdash \mathit{All}\ V\ f}\ (\text{All-I})\ [V \cap \bigcup(\mathit{Im}\ \mathit{FV}\ \Delta) = \emptyset] \qquad \frac{\Delta \vdash \mathit{All}\ V\ f}{\Delta \vdash f[\![\rho]\!]}\ (\text{All-E})\ [\mathit{Core}\ \rho \subseteq V]$$

Fig. 8. Natural deduction system for $\mathcal{L}_{\kappa_1,\kappa_2}$. Equality rules omitted—they are the same as for standard FOL.

also look at situations that violate equivariance (§9.3), and show that such situations can still benefit from strong induction with the price of being explicit about the involved binding structures (§9.4).

9.1 Example: Infinitary First-Order Logic

Given two infinite cardinals κ_1 and κ_2, the (κ_1, κ_2)-infinitary FOL logic $\mathcal{L}_{\kappa_1,\kappa_2}$ [Dickmann 1985; Keisler 1971; Marker 2016] is an extension of FOL which allows conjunctions / disjunctions of sets of formulas of any cardinality $< \kappa_1$, and quantifications over sets of variables of any cardinality $< \kappa_2$. ($\mathcal{L}_{\aleph_1,\aleph_0}$ is the best known version due to its importance for categorical logic [Makkai and Paré 1989].)

We let *Var* be an infinite set of cardinality $\kappa = \max(\kappa_1, \kappa_2)$. The set $\mathit{Fmla} = \mathit{Fmla}_{\kappa_1,\kappa_2}$ of $\mathcal{L}_{\kappa_1,\kappa_2}$-*formulas*, ranged over by f, is given by the grammar $f ::= \mathit{Eq}\ x\ y \mid \mathit{Neg}\ f \mid \mathit{Conj}\ F \mid \mathit{All}\ V\ f$ where F ranges (recursively) over $\mathcal{P}_{<\kappa_1}(\mathit{Fmla})$ (i.e., over sets of formulas of cardinality $< \kappa_1$) and V over $\mathcal{P}_{<\kappa_2}(\mathit{Var})$. Thus, a formula is either an equality, or a negation, or a conjunction over a set of formulas F, or a (simultaneous) quantification over a set of variables V. Again, formulas are identified modulo alpha-equivalence, e.g., $\mathit{All}\ \{x, y\}\ (\mathit{Eq}\ x\ y)$ and $\mathit{All}\ \{x, y\}\ (\mathit{Eq}\ y\ x)$ are the same formula.

Fig. 8 shows a straightforward generalization to $\mathcal{L}_{\kappa_1,\kappa_2}$ of the standard natural deduction rules for FOL, where Δ ranges over sets of formulas of cardinality $< \kappa$ and ρ over functions in $\mathit{Var} \to \mathit{Var}$. $\bot$ denotes the "false" formula, defined as $\mathit{Neg}\ (\mathit{Conj}\ \emptyset)$. Recall that $\mathit{Core}\ \rho$ denotes the core (support) of ρ, i.e., the set $\{x \in \mathit{Var} \mid \rho\ x \neq x\}$. Moreover, $\mathit{FV}\ f$ denotes the set of free variables of f, and $f[\![\rho]\!]$ denotes the (capture-free) parallel substitution of all free variables x in f with their ρ-image $\rho\ x$. The rules are standard except for accounting for the universal quantification of an entire set of variables V. Thus, the introduction rule (All-I) assumes freshness of all the variables in V for the hypotheses in Δ, and the elimination rule (All-E) makes sure that only variables in V are being instantiated.

By analogy with the finitary situations, we can hope to infer the following strong rule induction principle, which allows "avoiding" the bound variables V:

Prop 16. Let $(P, \mathit{Psupp} : P \to \mathcal{P}_{<\kappa}(\mathit{Var}))$ be a parameter structure. Let $\varphi : P \to \mathcal{P}_{<\kappa}\ \mathit{Fmla} \to \mathit{Fmla} \to \mathit{Bool}$ and assume that:

- [cases different from ⟪All-I⟫ and ⟪All-E⟫ omitted, as they don't involve binders]
- ⟪All-I⟫: $\forall p, \Delta, f.$
  $V \cap \mathit{Psupp}\ p = \emptyset \wedge V \cap \bigcup(\mathit{Im}\ \mathit{FV}\ \Delta) = \emptyset \wedge \Delta \vdash f \wedge (\forall q.\ \varphi\ q\ \Delta\ f) \longrightarrow \varphi\ \ p\ \Delta\ (\mathit{All}\ V\ f)$
- ⟪All-E⟫: $\forall p, \Delta, f.$
  $V \cap \mathit{Psupp}\ p = \emptyset \wedge \mathit{Core}\ \rho \subseteq V \wedge \Delta \vdash (\mathit{All}\ V\ f) \wedge (\forall q.\ \varphi\ q\ \Delta\ (\mathit{All}\ V\ f)) \longrightarrow \varphi\ p\ \Delta\ f$

Then $\forall p, \Delta, f.\ \Delta \vdash f \longrightarrow \varphi\ p\ \Delta\ f$.

9.2 An Infinitary Generalization of the Criterion

So how do we go about obtaining Prop. 16 from the inductive definition of deduction in Fig. 8? Everything seems to follow our usual pattern, except that *Fmla* is no longer a nominal set but only a "nominal-set-like" structure, where sets are not finite but bounded by a cardinal κ. We say that a set is κ-*small* if its cardinality is $< \kappa$; so $\mathcal{P}_{<\kappa}(X)$ is the set of its κ-small subsets of a set X. Let us call κ-*permutation* a bijection $\sigma : \mathit{Var} \to \mathit{Var}$ whose core is κ-small, and let Perm_κ denote the set of κ-

permutations. For formulas, the κ-permutation operator is here an action $\_[\_] : Fmla \to Perm_\kappa \to Fmla$ of $Perm_\kappa$ on $Fmla$; and the set of free variables $FV\ \varphi$ is no longer finite but only κ-small.

We therefore seek structures generalizing nominal sets in order to reach the following goals:

(G1) Infinitary syntaxes with static bindings and their permutation and free-variable operators, such as $(Fmla,\ \_[\_] : Fmla \to Perm_\kappa \to Perm_\kappa, FV : Fmla \to \mathcal{P}_{<\kappa}(Var))$ above, should form such structures—i.e., the support operator should give exactly the free variables.

(G2) Our strong rule induction criterion should carry over to these structures.

(G3) Ideally, these structures should be closed under relevant constructions (such as sums, products, container type extensions)—similarly to standard nominal sets.

However, the naive generalization of nominal sets to higher cardinalities κ, replacing "finite" with "κ-smallness", does not work. We sketch it below, in preparation for something that will actually work. Let us call *κ-pre-nominal set* any pair $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$ where $\_[\_]^{\mathcal{A}} : A \to Perm_\kappa \to A$ is an action on A of the monoid $(Perm_\kappa, 1_{Var}, \circ)$. Given a κ-pre-nominal set $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$, an $a \in A$ and a set $X \subseteq Var$, we define the notion of *X supports a* by adapting that from nominal sets: as $a[\sigma]^{\mathcal{A}} = a$ holding for all $\sigma \in Perm_\kappa$ such that $\forall a \in X.\ \sigma\ x = x$ (i.e., $X \subseteq Core\ \sigma$). Finally, we define a *κ-nominal set* to be a κ-pre-nominal set where every element has a κ-small supporting set. Now, the problem is that a fundamental property of nominal sets does not carry over to κ-nominal sets $\mathcal{A} = (A, \_[\_]^{\mathcal{A}})$ thus defined: Given $a \in A$, the least supporting set of a, which for nominal sets gave us the support $Supp^{\mathcal{A}}\ a$, is no longer guaranteed to exist. Here is a counterexample, which works for any $\kappa > \aleph_0$:

Counterexample 17. Let Var^∞ be the set of streams of variables. Given $xs \in Var^\infty$ and $i \in \mathbb{N}$, we write xs_i for the i'th variable in the stream. We say that two streams and ys are equivalent, written $xs \equiv ys$, if they are equal almost everywhere, i.e., there exists $n \in \mathbb{N}$ such that $xs_i = ys_i$ for all $i \geq n$. We let E be $Var^\infty/_\equiv$, the set of $\equiv$-equivalence classes. Given $xs \in Var^\infty$, we let $xs/_\equiv \in E$ denote its equivalence class. Since the standard permutation action on streams given by stream-map (so that $(xs[\sigma])_i = \sigma\ xs_i$ for each i) preserves $\equiv$, we can lift it to an operator on equivalence classes. This gives the κ-nominal set $\mathcal{E} = (E, \_[\_]^{\mathcal{E}})$ with $\_[\_]^{\mathcal{E}} : E \to Perm \to E$ defined as $(xs/_\equiv)[\sigma]^{\mathcal{E}} = (xs[\sigma])/_\equiv$. Now let $xs \in Var^\infty$ be any nonrepetitive stream. Each of the sets $\{x_i \mid i \geq n\}$ supports $xs/_\equiv$, but their intersection $\bigcap_{i \in \mathbb{N}} \{x_i \mid i \geq n\}$, which is empty, does not. So there is no least supporting set for $xs/_\equiv$.

Thus, if we switch from finite-core to κ-small-core permutations, we can no longer define the support as the least supporting set. But with goal (G2) in mind, we can ask whether our Thm. 7 really needs these least supporting sets or it can work with *any* supporting sets subject to weaker requirements. We discover these requirements looking back at Thm. 7's proof—where we have underlined the invocations of properties of the support operator $Supp = Supp^{\mathcal{T}}$ for the considered nominal set $\mathcal{T} = (T, \_[\_]^{\mathcal{T}})$. Fortunately, the minimality of $Supp$ is not needed in any of these. Rather:

- the last invocation of "properties of $Supp$" refers to the fact that $Supp$ returns supporting sets;
- the other invocations only require the property of the support being semi-natural w.r.t. permutation, in that $Supp\ (t[\sigma]) \subseteq Im\ \sigma\ (Supp\ t)$ for all $t \in T$ and $\sigma \in Perm$.

Thus, in the proof, we can replace the support operator with any operator satisfying the above two properties, which we will still call "support" (and denote by $Supp$). These more flexible assumptions allow a graceful transition from finiteness to κ-smallness. Indeed, our proof of Thm. 7 is resilient to this generalization as well: It only uses that finiteness is closed under permutation images and finite unions, which is also true about κ-smallness. This achieves goal (G2).

Remark 18. On the cardinality synchronization between support and permutations: For lifting the proof of Thm. 7 from finiteness to κ-smallness, it is essential that, in our generalization, permutations are allowed to "keep up" in cardinality with the support, in that the permutations now have κ-small cores (rather than just finite cores), matching the κ-smallness of the support. Indeed, the permutation

τ that we use in the proof to "refresh" the set B' for avoiding $Psupp\ p$ and $Supp\ (t[\sigma])$ (fact (vi)) must have its core's cardinality equal to that of B', which in the generalized version can be anything $< \kappa$.

Concerning goal (G1), it is easy to see that for the syntax of $\mathcal{L}_{\kappa_1,\kappa_2}$ (and any infinitary syntax for that matter), the free-variable operator FV satisfies the above desirable properties, in that $FV\ t$ is a supporting set for t and $FV\ (t[\sigma]) \subseteq Im\ \sigma\ (Supp\ t)$. We are therefore led to the following definition, in the context of a fixed infinite cardinal κ and a fixed set of variables Var such that $|Var| = \kappa$.

Def 19. A *κ-loosely-supported-nominal set* (*κ-LS-nominal set* for short) is a triple $\mathcal{A} = (A, \_[\_]^{\mathcal{A}}, Supp^{\mathcal{A}})$ where $\_[\_]^{\mathcal{A}} : A \to Perm_\kappa \to A$ and $Supp^{\mathcal{A}} : A \to \mathcal{P}_{<\kappa}(Var)$ are such that:
- $(A, \_[\_]^{\mathcal{A}})$ is a κ-pre-nominal set i.e., $\_[\_]^{\mathcal{A}}$ is an action of the monoid $(Perm_\kappa, \circ, 1_A)$ on A;
- $Supp^{\mathcal{A}}$ returns supporting sets, i.e., $(\forall x \in Supp^{\mathcal{A}}.\ \sigma\ x = x)$ implies $a[\sigma]^{\mathcal{A}}$ for all a and σ;
- $Supp^{\mathcal{A}}$ is *semi-natural*, i.e., $Supp^{\mathcal{A}}\ (a[\sigma]) \subseteq Im\ \sigma\ (Supp^{\mathcal{A}}\ a)$ for all a and σ.

The "loosely" qualifier refers to the support operator $Supp^{\mathcal{A}}$ no longer being "tied" to give a specific supporting set (the least one). Note that, thanks to the κ-pre-nominal set axioms, semi-naturality is actually equivalent to naturality: $Supp^{\mathcal{A}}\ (a[\sigma]) = Im\ \sigma\ (Supp^{\mathcal{A}}\ a)$ for all a and σ.

So Thm. 7 generalizes to κ-LS-nominal sets. We work with κ-LS-nominal sets $\mathcal{T} = (T, \_[\_]^{\mathcal{T}}, Supp^{\mathcal{T}})$ instead of nominal sets $\mathcal{T} = (T, \_[\_]^{\mathcal{T}})$, and the bound-variable argument B of the operator G is now in $\mathcal{P}_{<\kappa}(Var)$ rather than $\mathcal{P}_{\text{fin}}(Var)$. All the relevant notions, including equivariance and $\mathcal{T}$-refreshability, are defined like for nominal sets but replacing finiteness with κ-smallness.

Thm 20. Thm. 7 (also in its §8.3 strengthened form) still holds true if in its statement we replace:
- the nominal set $\mathcal{T} = (T, \_[\_]^{\mathcal{T}})$ and its support $Supp^{\mathcal{T}}$ with a κ-LS-nominal set $\mathcal{T} = (T, \_[\_]^{\mathcal{T}}, Supp^{\mathcal{T}})$;
- $G : (T \to Bool) \to (\mathcal{P}_{\text{fin}}(Var) \to T \to Bool)$ with $G : (T \to Bool) \to (\mathcal{P}_{<\kappa}(Var) \to T \to Bool)$;
- the parameter structure $(P, Psupp : P \to \mathcal{P}_{\text{fin}}\ (Var))$ with $(P, Psupp : P \to \mathcal{P}_{<\kappa}(Var))$.

So Thm. 20 (re)becomes Thm. 7 when $\kappa = \aleph_0$, and the κ-LS-nominal set $\mathcal{T} = (T, \_[\_]^{\mathcal{T}}, Supp^{\mathcal{T}})$ is a nominal set $\mathcal{T} = (T, \_[\_]^{\mathcal{T}})$ with its defined support operator. Moreover, when instantiating Thm. 20's operator G to that underlying the deduction system of $\mathcal{L}_{\kappa_1,\kappa_2}$, we obtain Prop. 16, as desired. Verifying the necessary hypotheses proceeds similarly to the finitary cases, via the §6 heuristic.

We have not yet addressed (G3), which bears upon the criterion's smooth instantiation, as it would allow constructing the required LS-nominal sets compositionally. It turns out that LS-nominal sets enjoy many of the closure properties of nominal sets [Pitts 2006; Urban 2008]. They are closed under the usual covariant set-theoretic (type-theoretic) constructions such as sums, products, and lifting via container types: both finitary ones such as lists, finite sets and bags, and infinitary ones such as streams, infinite trees, etc. (Our technical report [van Brügge et al. 2025b, App. C] gives details.)

In conclusion, we have extended our strong rule induction criterion to handle rule-based systems over infinitary structures with bindings, employing a mild extension of the nominal set axiomatization that still caters for concepts such as equivariance and refreshability. This should cover most of the infinitary situations of interest (including the ones cited at the beginning of §9). Our final stop in this paper is a case study where equivariance itself fails.

9.3 Example: Infinitary Affine λ-Calculus

In this subsection, Var will have cardinality $\aleph_1$, the first uncountable cardinal (so $\kappa = \aleph_1$). Recall that, A^∞ denotes the set of streams of elements in a set A, i.e. functions from $\mathbb{N}$ to A; we also let $A^{\infty,\neq}$ denote the subset of A^∞ consisting of the nonrepetitive streams, i.e., injective functions. Given $as \in A^\infty$, we write as_i for the i'th item in the stream, and $set\ as$ for the set of its elements $\{as_i \mid i \in \mathbb{N}\}$ (its image as a function). We let xs, ys etc. range over the set $Var^{\infty,\neq}$ of nonrepetitive streams of variables.

Following Mazza [2012], we define the syntax of *infinitary λ-calculus* by the following grammar, where t ranges over infinitary λ-terms (*λ-iterms*), i.e., elements of the syntax that is being introduced,

$$\mathit{affine}\ (\mathit{iVr}\ x)\ \text{(iVr)} \qquad \frac{\mathit{affine}\ t}{\mathit{affine}\ (\mathit{iLm}\ xs\ t)}\ \text{(iLm)} \qquad \frac{\begin{array}{c}\mathit{affine}\ t \\ \mathit{lift}\ (\lambda t'.\ \mathit{affine}\ t' \wedge \mathit{FV}\ t' \cap \mathit{FV}\ t = \emptyset)\ ts \\ \forall i, j.\ i \neq j \longrightarrow \mathit{FV}\ ts_i \cap \mathit{FV}\ ts_j = \emptyset\end{array}}{\mathit{affine}\ (\mathit{iAp}\ t\ ts)}\ \text{(iAp)}$$

Fig. 9. The *affine* predicate

and ts over streams of λ-iterms: $t ::= \mathit{iVr}\ x \mid \mathit{iAp}\ t\ ts \mid \mathit{iLm}\ xs\ t$. We assume that, in $\mathit{iLm}\ xs\ t$, the variables from the stream xs are bound in t; and λ-iterms are equated modulo the induced notion of alpha-equivalence. ILTerm denotes the set of λ-iterms. Given $t \in \mathit{ILTerm}$, $xs \in \mathit{Var}^{\infty,\neq}$ and $ts \in \mathit{ILTerm}^{\infty}$, we write $t[ts/xs]$ for the λ-iterm obtained by the simultaneous (capture-avoiding) substitution of the free occurrences in t of the variables xs_i with the corresponding λ-iterms ts_i.

Central in Mazza's development is the notion of a λ-iterm being *affine*, i.e., having no repeated occurrences of any free variable in it, or in any of its subterms (including subterms located under binders). This is expressed by the inductive predicate $\mathit{affine} : \mathit{ILTerm} \to \mathit{Bool}$ from Fig. 9, to which our Thm. 20 instantiates seamlessly, yielding the following strong induction principle. (Since $\kappa = \aleph_1$, $\mathcal{P}_{<\kappa}(\mathit{Var})$ is $\mathcal{P}_{\mathsf{countable}}(\mathit{Var})$, the set of countable subsets of Var.)

Prop 21. Let $(P, \mathit{Psupp} : P \to \mathcal{P}_{\mathsf{countable}}(\mathit{Var}))$ and $\varphi : P \to \mathit{ILTerm} \to \mathit{Bool}$, and assume that:
- [cases different from ⦇iLm⦈ omitted, as they don't involve binders]
- ⦇iLm⦈: $\forall p, xs, t.\ \mathit{set}\ xs \cap \mathit{Psupp}\ p = \emptyset \wedge \mathit{affine}\ t \wedge (\forall q.\ \varphi\ q\ t) \longrightarrow \varphi\ p\ (\mathit{iLm}\ xs\ t)$

Then $\forall p, t.\ \mathit{affine}\ t \longrightarrow \varphi\ p\ t$.

Since in our criterion the rules' hypotheses are not required to fit any syntactic format, higher-order operators and quantifiers such as Fig. 9's *lift* (which lifts a predicate from elements to streams, i.e., is defined by $\mathit{lift}\ \varphi\ as = (\forall i \in \mathbb{N}.\ \varphi\ as_i)$) can be used freely.

Mazza [2012]'s goal is to establish an isomorphic translation between (finitary) λ-calculus and a suitably *uniform* version of affine infinitary λ-calculus. This maps an application λ-term $\mathit{Ap}\ s\ t$ to an application λ-iterm $\mathit{iAp}\ s'\ ts'$, where s' is (recursively) an infinitary counterpart of s and ts' is a stream of copies of infinitary counterparts of t, with the copies having disjoint variables but otherwise having the same structure; and maps an abstraction λ-term $\mathit{Lm}\ x\ t$ to an abstraction λ-iterm $\mathit{Lm}\ xs'\ t'$, where t' is an infinitary counterpart of t and xs' is a nonrepetitive stream of copies of x.

To describe the image of this translation, Mazza fixes a countable subset $\mathit{Super} \subseteq \mathit{Var}^{\infty,\neq}$ of nonrepetitive streams of variables called *supervariables*, having the property that any two are mutually disjoint: $\forall xs, ys \in \mathit{Super}.\ \mathit{set}\ xs \cap \mathit{set}\ ys = \emptyset$. The intention is restricting the λ-iterms to only use these as bindings. Namely, supervariables induce the notion of *renaming equivalence* expressed as the relation $\approx : \mathit{ILTerm} \to \mathit{ILTerm} \to \mathit{Bool}$ which relates two λ-iterms t and t' just in case they **(1)** have the same $(\mathit{iVr}, \mathit{iLm}, \mathit{iAp})$-structure (as trees), **(2)** only use supervariables in binders, **(3)** at the leaves have variables appearing in the same supervariable, and **(4)** for both t and t' all the subterms that form the righthand side of an application are mutually renaming equivalent. The $\approx$ relation is defined inductively in Fig. 10, via rules having a logical relation flavor. (Then *uniformity* of an λ-iterm, which together with affineness characterizes the translation's image, is defined as that λ-iterm being renaming-equivalent to itself. Our technical report [van Brügge et al. 2025b, App. E] gives details.)

Note that the set *Super* is not guaranteed to be closed under permutation. Even worse, it actually cannot be chosen so that it is closed, due to the disjointness assumption: If we permute some variables in a supervariable xs we obtain a stream of variables that is distinct but not disjoint from xs, which therefore cannot be a supervariable. For this reason, the monotonic operator underlying the definition of $\approx$, and the relation $\approx$ itself, are hopelessly non-equivariant, which renders strong induction impossible via current nominal criteria, including our own LS-nominal one. However, intuition tells us that when inducting over $\approx$ we should still be able to avoid the bound variables

$$\frac{xs \in Super \quad \{x, x'\} \subseteq set\ xs}{iVr\ x\ \approx\ iVr\ x'}\ (\text{iVr}) \qquad \frac{xs \in Super \quad t \approx t'}{iLm\ xs\ t\ \approx\ iLm\ xs\ t'}\ (\text{iLm}) \qquad \frac{\begin{array}{c} t \approx t' \\ \forall t_1, t_2.\ \{t_1, t_2\} \subseteq set\ ts \cup set\ ts' \\ \longrightarrow t_1 \approx t_2 \end{array}}{iAp\ t\ ts\ \approx\ iAp\ t'\ ts'}\ (\text{iAp})$$

Fig. 10. Mazza's renaming equivalence relation

$$\begin{array}{rcl} G\ \varphi\ b\ (s, s') & \Longleftrightarrow & \begin{array}{ll} (1) & (\exists xs, x, x'.\ b = \bot \wedge s = iVr\ x \wedge s' = iVr\ x' \wedge xs \in Super \wedge \{x, x'\} \subseteq set\ xs)\ \vee \\ (2) & (\exists xs, t, t'.\ \varphi\ (t, t')\ \wedge b = xs \wedge s = iLm\ xs\ t \wedge s' = iLm\ xs\ t' \wedge xs \in Super)\ \vee \\ (3) & (\exists t, ts, t', ts'.\ \varphi\ (t, t') \wedge (\forall t_1, t_2.\ \{t_1, t_2\} \subseteq set\ ts \cup set\ ts' \\ & \qquad \longrightarrow \varphi\ (t_1, t_2))\ \wedge b = \bot \wedge s = iAp\ t\ ts \wedge s' = iAp\ t'\ ts') \end{array} \end{array}$$

Fig. 11. The operator associated to renaming equivalence

xs, similarly to how we did for *affine*, provided the parameters do not stretch too wide w.r.t. supervariables. And a reasonable notion of not stretching too wide is *touching only finitely many supervariables*. Thus, we can hope for the following strong induction principle for ≈, where the first highlighted part formalizes this condition regarding supervariables:

Prop 22. Let $(P, Psupp : P \to \mathcal{P}_{\text{countable}}(Var))$ be such that, for any $p \in P$, $\{xs \in Super \mid set\ xs \cap Psupp\ p \neq \emptyset\}$ is finite. Let $\varphi : P \to ILTerm \to ILTerm \to Bool$ and assume the following:

- [cases different from ⟪iLm⟫ omitted, as they don't involve binders]
- ⟪iLm⟫: $\forall p, xs, t, t'.\ set\ xs \cap Psupp\ p = \emptyset \wedge xs \in Super \wedge t \approx t' \wedge (\forall q.\ \varphi\ q\ t\ t') \longrightarrow \varphi\ p\ (iLm\ xs\ t)\ (iLm\ xs\ t')$

Then $\forall p, t, t'.\ t \approx t' \longrightarrow \varphi\ p\ t\ t'$.

9.4 A Criterion with Explicit Binders

The more general question we are led to is: *Can we still obtain strong induction in situations where equivariance fails, namely in the presence of non-equivariant restrictions on binders (such as the above supervariable restriction)?* To answer this, our Thm. 20's (and Thm. 7's) blurred view of binders needs to be sharpened. Indeed, the theorem refers to an inductive predicate's underlying operator G that acts not on binders directly, but on sets B of variables that are typically obtained by collecting the variables bound in the rules' conclusions; e.g., for the (iLm) rule for *affine* in Fig. 9, B is *set xs*. However, the set of variables in a binder can be oblivious to restrictions on binders, as is the case with supervariables in the ≈ example: two streams, one in and one not in *Super*, can have the same set of variables. Thus, when dealing with non-equivariant restrictions on binders, we must consider binders as first-class citizens. And LS-nominal sets again come handy for modeling this.

In addition to the κ-LS-nominal set of term-like items $\mathcal{T} = (T, \_[\_]^{\mathcal{T}}, Supp^{\mathcal{T}})$ (as before), we consider another κ-LS-nominal set $\mathcal{B} = (B, \_[\_]^{\mathcal{B}}, Supp^{\mathcal{B}})$ of items that we will call "binders", and an operator $G : (T \to Bool) \to (B \to T \to Bool)$. Provided G is monotonic, we again iterate it to define the predicate $I_G : T \to Bool$ inductively by the rule $\frac{G\ I_G\ b\ t}{I_G\ t}$. To tackle the problem with non-equivariance, the key is to identify a suitable notion of *relative* equivariance, subject to sanity conditions w.r.t. freshness. We fix a predicate $bnd : B \to Bool$ that singles out certain binders that are well-formed w.r.t. our considered inductive definition, and define $Perm_{\kappa,bnd}$ to be the set of κ-permutations $\sigma : Var \to Var$ that, applied via $\_[\_]^{\mathcal{B}}$, preserve well-formedness of binders, in that $\forall b \in B.\ bnd\ b \longrightarrow bnd\ (b[\sigma]^{\mathcal{B}})$. And we define *bnd-equivariance* by restricting equivariance to the bijections in $Perm_{\kappa,bnd}$. For example, a predicate $\varphi : T \to Bool$ is *bnd*-equivariant when $\varphi\ t$ implies $\varphi\ (t[\sigma]^{\mathcal{T}})$ for all $t \in T$ and $\sigma \in Perm_{\kappa,bnd}$.

We correspondingly generalize weak $\mathcal{T}$-refreshability: G is called is *weakly* $(\mathcal{T}, \mathcal{B}, bnd)$*-refreshable* when, for all $\varphi : T \to Bool$, $b \in B$ and $t \in T$, if $\forall t \in T.\ \varphi\ t \longrightarrow I_G\ t$, φ is *bnd*-equivariant and $G\ \varphi\ b\ t$,

then there exists $b' \in B$ with $Supp^{\mathcal{B}}\ b \cap Supp^{\mathcal{T}} t = \emptyset$ and $G\ \varphi\ b'\ t$. Moreover, G is said to be *bnd-compatible* if it only holds for items satisfying the *bnd* restriction: $G\ R\ b\ t$ implies $bnd\ b$ for all R, b, t.

Finally, we want to be able express notions of size for our explicit binders that go beyond mere cardinality, such as "touching only finitely many supervariables". Rather than attempting to get too specific here, we employ an abstract predicate $bsmall : \mathcal{P}(Var) \to Bool$ (read "binder-small") subject to some sanity assumptions: *bsmall* is said to be *closed under union* if $bsmall\ X$ and $bsmall\ Y$ implies $bsmall\ (X \cup Y)$ for all $X, Y \subseteq Var$. Moreover, I_G and *bnd* are said to be *bsmall-compatible* if $I_G\ t$ implies $bsmall\ (Supp^{\mathcal{T}} t)$ for all $t \in T$, and $bnd\ b$ implies $bsmall\ (Supp^{\mathcal{B}}\ b)$ for all $b \in B$, respectively.

The above generalizes our previous setting for strong rule induction, which can be obtained by taking $\mathcal{B}$ to be the κ-LS-nominal set having $B = \mathcal{P}_{<\kappa}(Var)$, and $\_[\_]^{\mathcal{B}}$ and $Supp^{\mathcal{B}}$ as the image and identity operators, respectively; and taking *bnd* and *bsmall* to be vacuously true.

All the above assumptions should be expected to hold for most reasonable choices of the *bsmall* predicate, and indeed they hold for the $\approx$ example if we take *bsmall* to mean "touches only finitely many supervariables". So we can hope for a strong induction theorem that works when further restricting the parameters with a *bsmall*-ness assumption. And indeed, the proof of Thm. 20 (which was in turn adapted from that of Thm. 7) almost works, save for the step where we proved the existence of a permutation τ such that the facts labelled (vi) and (vii) hold. We want something similar to the cardinality reasoning invoked there, which applies to κ-smallness, to also apply to binder-smallness. We call the predicate *bnd bsmall-liftable* when the following condition holds: For all $A, A' \in \mathcal{P}_{<\kappa}(Var)$ and $b \in B$ such that $bsmall\ A$ and $bsmall\ A'$, if $A' \subseteq A$ and $Supp^{\mathcal{B}}\ b \cap A' = \emptyset$, then there exists $\tau \in Perm_{\kappa,bnd}$ such that $Im\ \tau\ (Supp^{\mathcal{B}}\ b) \cap A = \emptyset$ and $\forall x \in A'.\ \tau\ x = x$.

With these ingredients, we can prove a binder-explicit strong rule induction criterion. A parameter structure $\mathcal{P} = (P, Psupp : P \to \mathcal{P}_{<\kappa}(Var))$ is called *bsmall-compatible* if $bsmall\ (Psupp\ p)$ for all p.

Thm 23. Let $\mathcal{T} = (T, \_[\_]^{\mathcal{T}}, Supp^{\mathcal{T}})$ and $\mathcal{B} = (B, \_[\_]^{\mathcal{B}}, Supp^{\mathcal{B}})$ be κ-LS-nominal sets, $bnd : B \to Bool$ and $bsmall : \mathcal{P}(Var) \to Bool$ predicates, and $G : (T \to Bool) \to (B \to T \to Bool)$ an operator, such that: **(1)** G is monotonic, *bnd*-compatible, *bnd*-equivariant and $(\mathcal{T}, \mathcal{B}, bnd)$-refreshable; **(2)** *bsmall* is closed under union; **(3)** I_G and *bnd* are *bsmall*-compatible; **(4)** *bnd* is *bsmall*-liftable.

Let $(P, Psupp)$ be a *bsmall*-compatible parameter structure and $\varphi : P \to T \to Bool$ such that:

$$\forall p \in P, t \in T, b \in B. \left(\begin{array}{c} Supp^{\mathcal{B}}\ b \cap (Psupp\ p \cup Supp^{\mathcal{T}} t) = \emptyset \ \wedge \\ G\ (\lambda t'.\ I_G\ t' \wedge \forall p' \in P.\ \varphi\ p'\ t')\ b\ t \end{array} \right) \longrightarrow \varphi\ p\ t.$$

Then $\forall p \in P, t \in T.\ I_G\ t \longrightarrow \varphi\ p\ t$.

Thm. 23 is, by design, a generalization of Thm. 20. Also, it can be instantiated to obtain the desired strong induction for renaming equivalence, namely Prop. 21 (taking $\kappa = \aleph_1$):

- $\mathcal{T} = (T, \_[\_]^{\mathcal{T}}, Supp^{\mathcal{T}})$ taken as the $\aleph_1$-LS-nominal set structure on $T = \mathit{LTerm}^2$;
- $\mathcal{B} = (B, \_[\_]^{\mathcal{B}}, Supp^{\mathcal{B}})$ defined by taking $B = Var_{\perp}^{\infty,\neq} = Var^{\infty,\neq} \cup \{\perp\}$, where the elements of $Var^{\infty,\neq}$ are the proper binders and $\perp$ means "no binder"; and taking $\_[\_]^{\mathcal{B}}$ and $Supp^{\mathcal{B}}$ as the liftings to $Var_{\perp}^{\infty,\neq}$ of the map and set operators from $Var^{\infty,\neq}$;
- *bnd* defined to hold for $\perp$ and for any $xs \in Super$;
- *bsmall* A defined as "$\{xs \in Super \mid set\ xs \cap A \neq \emptyset\}$ finite";
- G as shown in Fig. 11, making I_G (the uncurried version of) $\approx$.

The verification of Thm. 23's $\mathcal{T}$-refreshability assumption goes by a straightforward variation of our previous heuristic, working with permutations applied directly to binders (via $\_[\_]^{\mathcal{B}}$) rather than to sets of bound variables (via *Im*). Moreover, the *bnd*-compatibility of G, the closedness of *bsmall* under union, and the *bsmall*-compatibility of *bnd* are immediate; and the *bsmall*-compatibility of I_G follows by routine standard induction on I_G. The only non-routine check is that of the *bsmall*-liftability of *bnd*, which amounts to the following property: For all $xs \in Super$ and countable sets of variables A, A' that touch only finitely many supervariables and such that $A' \subseteq A$ and A'

does not touch xs, there exists a supervariable-preserving permutation σ on variables such that $\{\sigma\ x \mid x \in set\ xs\} \cap A = \emptyset$ and $Core\ \sigma \cap A' = \emptyset$. This is proved by choosing a supervariable ys that is distinct (hence disjoint) from xs and is not touched by A, and defining τ to swap the elements of xs and ys componentwise and to be identity everywhere else (hence on A' too).

10 Tool Support and Case Studies in Isabelle/HOL

We mechanized this paper's general theorems, instances and (counter) examples in Isabelle/HOL [Nipkow et al. 2002]. We further validated our induction principles in two proof developments: transitivity of the System $F_{<:}$ subtyping (part of POPLmark [Aydemir et al. 2005]) and the isomorphism between the affine uniform infinitary λ-calculus and the standard λ-calculus [Mazza 2012]. (Our technical report [van Brügge et al. 2025b, Apps. E, F, and G.3] gives details.) We also pursued an abstract case study: proving the rule-format based criterion of Urban et al. [2007] as an instance of our theorem. (Our technical report [van Brügge et al. 2025b, App. A] gives details.)

To support the use of the general theorems in concrete instances, we implemented a definitional extension of Isabelle's inductive specification and proof facilities, exported to users as new commands binder_datatype, binder_inductive, and make_binder_inductive, and the proof method binder_induction. The implementation and mechanization are available [van Brügge et al. 2025a].

From a user specification of the syntax and its binders, the command binder_datatype defines the type of terms for that syntax quotiented to alpha-equivalence along the foundations sketched by Blanchette et al. [2019]. It also defines the constructors, renaming and free variable operators, proves their basic properties, and infers structural induction and recursion principles. We deployed it to obtain all this paper's datatypes: λ-terms, π-calculus processes and commitments, System $F_{<:}$ types, $\mathcal{L}_{\kappa_1,\kappa_2}$-formulas, and λ-iterms. (Our technical report [van Brügge et al. 2025b, Apps. G.1 and D] gives details.)

Our general rule induction criteria, Thms. 7, 20 and 23, were formalized using Isabelle's locales [Ballarin 2014; Kammüller et al. 1999], a module system allowing to fix parameters, make assumptions about them, and infer consequences from these assumptions. For example, with Thm. 20 the parameters are the tuple $\mathcal{T} = (T, \_[\_]^{\mathcal{T}}, Supp^{\mathcal{T}})$ and the operator $G : (T \to Bool) \to (\mathcal{P}_{<\kappa}(Var) \to T \to Bool)$, the assumptions are that $\mathcal{T}$ is a κ-LS-nominal set and G is monotonic, equivariant and (weakly) $\mathcal{T}$-refreshable; and the culmination of what is being inferred in that locale is the conclusion of Thm. 20, i.e., that the indicated strong rule induction holds for the predicate I_G defined inductively from G. Similarly for Thm. 7 and Thm. 23. Since Thm. 23 is more general than Thm. 20 which in turn is more general than Thm. 7, we only proved Thm. 23 directly and inferred Thm. 20 by showing how the former's parameters and assumptions can be instantiated to the latter's parameters and assumptions via a *sublocale* relationship (and similarly for inferring Thm. 7 from Thm. 20). Results stated in a locale can be obtained by *interpretation*, Isabelle's mechanism for instantiating a locale's parameters with concrete values and discharging the assumptions.

The commands binder_inductive and make_binder_inductive provide a high-level language for the user to endow an inductive predicates with a strong (binding-aware) rule induction principle (as an instance of our general result). binder_inductive behaves like the Isabelle/HOL inductive command for specifying standard inductive predicates by instantiating the Knaster-Tarski theorem (a command available in most HOL-based provers), but it additionally attempts to formulate and prove a strong rule induction principle. Namely, from a user specification of such a predicate using syntax identical to that required by the inductive command, our tool derives the relevant nominal set (or κ-LS-nominal set) infrastructure and the low-level operator G (as shown in this paper's examples), proves an instance of Thm. 20 for G, and outputs the strong induction theorem and other useful results such as the inductive predicate's equivariance. Currently, the tool automates the proofs of the (κ-LS-)nominal set axioms and equivariance, but requires the user to prove $\mathcal{T}$-refreshability—

typically following the heuristic described in Section 6, which we have supported via some Isabelle tactics. The command make_binder_inductive is an incremental alternative to binder_inductive, allowing to decouple the standard inductive definition of a predicate (via "inductive") from its registration to produce a strong rule induction principle for it. Thus, issuing binder_inductive is equivalent to issuing an "inductive" followed by make_binder_inductive. The advantage of this decoupled approach is that in between the "inductive" and make_binder_inductive commands one can state and prove any inductive properties of the predicate needed in the proof of the assumptions for strong rule induction (as illustrated at the end of §8.2). The concrete strong rule induction priniciples for most examples (Props. 2, 13, 14, 16, 21, and all others mentioned in our technical report [van Brügge et al. 2025b, Apps. B and F]) were obtained using `binder_inductive`. The strong rule induction principles requiring explicit binders (Thm. 23) such as Prop. 22 and others from our technical report [van Brügge et al. 2025b, App. E] were obtained by manual locale interpretation.

Finally, our proof method binder_induction makes strong induction convenient to deploy in proofs. It allows the users to start induction while indicating the parameters to be avoided, as opposed to building the parameter structure explicitly.

We conclude with an example of our toolbox for the working syntax-with-bindings formalizer in action: the declaration of the datatype of System $F_{<:}$ types, the subtyping relation, and an example proof outline (of weakening of subtyping) with essential elements particular to our tools highlighted, namely the binding information for the datatype's constructors—here, the fact that the Forall constructor (denoted by ∀ in §8.2) binds the first (variable) argument into the third argument, and the parameters to be avoided when applying strong rule induction to prove weakening.

```
binder_datatype 'tvar sftypeP = TVr 'tvar | Top | Fun ('var sftypeP) ('var sftypeP)
    | Forall (x::'tvar) ('tvar typ) (t::'tvar typ) binds x in t

type_synonym sftype = tvar sftypeP

inductive ty :: (tvar × sftype) list ⟶ tvar sftype ⟶ tvar sftype ⟶ bool (_ ⊢ _ <: _) where
  SA_Top:        wf Γ ⟹ closed_in S Γ ⟹ Γ ⊢ S <: Top
| SA_Refl_TVar:  wf Γ ⟹ closed_in (TyVar x) Γ ⟹ Γ ⊢ TyVar x <: TyVar x
| SA_Trans_TVar: (x, U) ∈ set Γ ⟹ Γ ⊢ U <: T ⟹ Γ ⊢ TyVar x <: T
| SA_Arrow:      Γ ⊢ T₁ <: S₁ ⟹ Γ ⊢ S₂ <: T₂ ⟹ Γ ⊢ Fun S₁ S₂ <: Fun T₁ T₂
| SA_All:        Γ ⊢ T₁ <: S₁ ⟹ Γ;(x,T₁) ⊢ S₂ <: T₂ ⟹ Γ ⊢ Forall x S₁ S₂ <: Forall x T₁ T₂

⋮ 2 immediate lemmas about typing (mentioned at the end of §8.2) proved by rule induction

make_binder_inductive ty

    ⋮ 30 lines proof of weak 𝒯-refreshability using the heuristic (§6)

lemma ty_weakening: ⟦Γ ⊢ S <: T; ⊢ wf (Γ;Δ)⟧ ⟹ Γ;Δ ⊢ S <: T
proof (binder_induction Γ S T avoiding: dom Δ rule: ty_strong_induct)

    ⋮ 12 lines routine proof using the strong induction principle's Barendregt convention
```

Note that our datatype 'var sftypeP for System $F_{<:}$ types is polymorphic in the type 'tvar of (type) variables—and this is the case with all our datatypes for this paper's examples. This is to achieve slightly higher generality. Namely, instead of working with a fixed set of variables of suitable cardinality (which in the finitary case is just $\aleph_0$), that set is kept as a parameter—and in Isabelle/HOL, taking advantage of polymorphism, this is a type variable 'tvar of type class that specifies the cardinality constraint. (The binder_datatype command automatically assigns 'tvar to have the suitable type class.) This allows more flexibility in case we want to nest the given datatype inside another datatype that perhaps requires larger sets of variables. But once the exact datatypes needed

for a case study have been decided, to cut down the unnecessary polymorphism we instantiate the type variables with fixed types; here, we instantiate 'tvar with a fixed type tvar of suitable cardinality (here, countable), and sftype is introduced as an Isabelle type synonym for tvar sftypeP, i.e., for the instance of the polymorphic type 'tvar sftypeP with the fixed type tvar. The subsequent inductive and make_binder_inductive commands shown above use this monomorphic type.

11 Further Related Work

The proximal related work is Urban et al. [2007], which we have extensively discussed throughout this paper. It is the literature's most general account of rule induction obeying Barendregt's variable convention. Its syntactic format criterion generalizes previous others, which operate on particular syntaxes [Bengtson 2010; McKinna and Pollack 1999; Norrish 2006; Urban and Norrish 2005].

Complementary to our work on binding-aware rule induction is work on binding-aware datatypes. This includes general mechanisms for building alpha-quotiented datatypes for binding signatures [Blanchette et al. 2019; Pitts 2006; Urban and Kaliszyk 2012], and also Barendregt-convention observing (strong) structural induction and recursion [Blanchette et al. 2019; Norrish 2004; Pitts 2006]. Since structural induction can be regarded as a particular case of rule induction (for the monotonic operator that applies the datatype's constructors), our work can be seen as generalizing the strong induction components of those works—although the main difficulty there lies with the construction of the datatypes and the inference of the recursion principles, which are orthogonal to our contribution.

Our tool described in §10 provides support for both binding-aware rule induction and binding-aware datatypes in Isabelle. It is more expressive than Nominal Isabelle [Urban and Tasson 2005] (including the Nominal 2 variant [Urban and Kaliszyk 2012]) in both the allowed datatypes and inductive predicates—reflecting the higher generality and flexibility of our criterion compared to Urban et al. [2007]. But it is currently in a prototype stage, lacking Nominal Isabelle's high degree of automation which has been finetuned based on feedback from its many users over the years. We are contemplating a future integration of these two tools, combining the best of both worlds.

We are not the first to relax the finite support assumption of nominal sets—Pitts [2013, §2.10] summarizes existing approaches. On the way to his completeness theorem for nominal logic, Cheney [2006] generalizes the support operator by noticing that the finite subsets of atoms (in our terminology, variables) $\mathcal{P}_{\mathsf{fin}}(Var)$ form an ideal of $\mathcal{P}(Var)$ that contains all singleton sets $\{x\}$, and replacing $\mathcal{P}_{\mathsf{fin}}(Var)$ with an arbitrary such ideal $\mathcal{I}$, thus introducing $\mathcal{I}$-nominal sets—defined as pre-nominal sets such that every element has a supporting set of atoms from $\mathcal{I}$. The role of $\mathcal{I}$-nominal sets is that nominal logic deduction becomes complete w.r.t. these looser, ideal-supported models. Since $\mathcal{P}_{<\kappa}(Var)$ is also such an ideal, Cheney's $\mathcal{I}$ -nominal sets cover structures with infinite support. However, regardless of the ideal $\mathcal{I}$ (be it $\mathcal{P}_{\mathsf{fin}}(Var)$, or $\mathcal{P}_{<\kappa}(Var)$, etc.), Cheney still defines the notion of supporting set using swapping, which is equivalent to using *finite*-core permutations, whereas we allow larger permutations whose cores have cardinality $< \kappa$. While staying with finite-core permutations was suitable for Cheney's goal of proving completeness, as we discuss in Remark 18 strong induction coping with κ-small support requires κ-small-core permutations. Since $\mathcal{I}$-nominal sets are (semi-)natural w.r.t. finite-core permutations only, a variation of our Thm. 20 would apply to $\mathcal{I}$-nominal sets if we restricted the parameters to be finitely supported (i.e., *Psupp* to return finite sets). But being able to avoid only finitely many variables when proving facts about structures having infinitely many (free) variables would not be very useful.

Dowek and Gabbay [2012] introduce *permissive nominal sets*, a generalization of nominal sets based on separating atoms (variables) in two categories, along the distinction between free and bound variables. The elements in permissive nominal sets have supporting *permission sets*, which contain finitely many atoms of one category and co-finitely many of the other; this ensures the existence of least supporting sets. Like with Cheney's $\mathcal{I}$-nominal sets and differently from our

κ-LS-nominal sets, the notion of supporting set is defined there using finite-core permutations. Permissive nominal sets are the semantic underpinning of *permissive nominal logic* [Dowek and Gabbay 2012, 2023; Dowek et al. 2010], an elaborate extension of nominal logic with enhanced support for contextual and higher-order reasoning. Another difference between both $\mathcal{I}$-nominal sets and permissive nominal sets and our LS-nominal sets is that, the former retain the minimality of the support whereas in the latter we replaced minimality with the weaker axiom of (semi-)naturality.

Gabbay [2007] develops a nominal-style axiomatic set theory, FMG (Fraenkel-Mostowski Generalized), which generalizes the Fraenkel-Mostowski set theory previously introduced by Gabbay and Pitts [2002] as a foundation for nominal logic. In FMG, "smallness" of a set (such as the support of an item) no longer means "finiteness", but the possibility to internally well-order that set. This covers in particular cardinality bounds like the ones we use in LS-nominal sets. When developing his theory, Gabbay also constructs datatypes and develops mechanisms for extending functions from representatives to equivalence classes (via his *Barendregt abstractive* functions). Our preliminary investigations suggest that our criterion for strong rule induction could be adapted to Gabbay's FMG, complementing his results about datatypes and recursive-function definition principles.

Like the above works, we operate within (a transfinite generalization of) the nominal paradigm, where the names of the variables are visible, but ultimately irrelevant in that their choice does not matter. Barendregt's convention only makes sense in this paradigm. The other two major paradigms on representing and reasoning about syntax with bindings are based on nameless / De Bruijn representations [de Bruijn 1972] (and its type-safe and scope-safe generalizations, e.g., [Allais et al. 2018; Fiore et al. 1999; Schäfer et al. 2015]) and higher-order abstract syntax (HOAS) [Baelde et al. 2014; Harper et al. 1987; Pfenning and Elliott 1988; Pfenning and Schürmann 1999; Pientka 2010]. (Cross-paradigm hybrids have also been proposed, e.g., [Aydemir et al. 2008; Charguéraud 2012; Felty and Momigliano 2012; McKinna and Pollack 1999; Pollack et al. 2012].) There are relative pros and cons between these paradigms [Abel et al. 2017; Berghofer and Urban 2006; Felty and Momigliano 2012; Gheri and Popescu 2020; Kaiser et al. 2017; Norrish and Vestergaard 2007]. An advantage of the nominal paradigm is faithfulness to the informal, textbook descriptions of the systems. Our contribution is also in this direction, by lowering the informal-formal gap in nominal-style strong rule induction.

While our LS-nominal sets accommodate both infinitely branching and infinitely deep (non-well-founded) syntax, our infinitary examples (in §9.1, §9.3, and our technical report [van Brügge et al. 2025b, App. E]) only involve the former. The latter also has a rich literature—centered around concepts such as Böhm, Lévy-Longo and Berarducci trees [Barendregt and Klop 2009; Berarducci and Dezani-Ciancaglini 1999], used in the λ-calculus semantics. While inductively defined predicates on non-well-founded trees will fall under our strong induction criterion, such structures are often best explored not inductively, but coinductively, i.e., via predicates defined not as least but as greatest fixed points. We leave as future work the study of Barendregt's variable convention for rule-based coinduction. This would complement existing results on nominal-syle codatypes and corecursion [Blanchette et al. 2019; Kurz et al. 2012, 2013; Milius and Wißmann 2015; Popescu 2024].

Acknowledgments

We are grateful to the paper reviewers for their excellent questions and suggestions for improvement, which we were able to factor in thanks to the ample additional space available for the final version; and to the paper and artifact reviewers for identifying some far-reaching typos. Popescu acknowledges support from the EPSRC grant EP/X015114/1, titled "Safe and secure COncurrent programming for adVancEd aRchiTectures (COVERT)". Traytel acknowledges support from the Novo Nordisk Foundation (start package grant NNF20OC0063462) and the Independent Research Fund Denmark (DFF Sapere Aude grant 3120-00057B, titled DISCOVER). The authors are listed alphabetically regardless of individual contributions or seniority.

References

2024. The HOL4 Theorem Prover. http://hol.sourceforge.net/.

Andreas Abel, Alberto Momigliano, and Brigitte Pientka. 2017. POPLMark Reloaded. In *Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP) 2017*, Marino Miculan and Florian Rabe (Eds.). https://lfmtp.org/workshops/2017/inc/papers/paper_8_abel.pdf

Guillaume Allais, Robert Atkey, James Chapman, Conor McBride, and James McKinna. 2018. A Type and Scope Safe Universe of Syntaxes with Binding: Their Semantics and Proofs. *Proc. ACM Program. Lang.* 2, International Conference on Functional Programming (ICFP) (2018), 90:1–90:30. http://doi.acm.org/10.1145/3236785

Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. 2005. Mechanized Metatheory for the Masses: The PoplMark Challenge. In *Theorem Proving in Higher Order Logics (TPHOLs) 2005*, Joe Hurd and Thomas F. Melham (Eds.). LNCS, Vol. 3603. Springer, 50–65. https://doi.org/10.1007/11541868_4

Brian E. Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. 2008. Engineering Formal Metatheory. In *Principles of Programming Languages (POPL) 2008*, George C. Necula and Philip Wadler (Eds.). ACM, 3–15. https://doi.org/10.1145/1328438.1328443

David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. 2014. Abella: A System for Reasoning about Relational Specifications. *J. Formalized Reasoning* 7, 2 (2014), 1–89. https://doi.org/10.6092/issn.1972-5787/4650

Clemens Ballarin. 2014. Locales: A Module System for Mathematical Theories. *J. Autom. Reason.* 52, 2 (2014), 123–153. https://doi.org/10.1007/s10817-013-9284-7

Henk Barendregt and Jan Willem Klop. 2009. Applications of infinitary lambda calculus. *Inf. Comput.* 207, 5 (2009), 559–582. https://doi.org/10.1016/J.IC.2008.09.003

Hendrik Pieter Barendregt. 1985. *The lambda calculus - its syntax and semantics*. Studies in logic and the foundations of mathematics, Vol. 103. North-Holland.

Jesper Bengtson. 2010. *Formalising process calculi*. Ph. D. Dissertation. Uppsala University, Sweden. http://www.itu.dk/people/jebe/files/thesis.pdf

Jesper Bengtson. 2012. The pi-calculus in nominal logic. *Arch. Formal Proofs* 2012 (2012). https://www.isa-afp.org/entries/Pi_Calculus.shtml

Alessandro Berarducci and Mariangiola Dezani-Ciancaglini. 1999. Infinite lambda-Calculus and Types. *Theor. Comput. Sci.* 212, 1-2 (1999), 29–75. https://doi.org/10.1016/S0304-3975(98)00135-2

Stefan Berghofer and Christian Urban. 2006. A Head-to-Head Comparison of de Bruijn Indices and Names. In *LFMTP 2006 (ENTCS, Vol. 174)*, Alberto Momigliano and Brigitte Pientka (Eds.). Elsevier, 53–67. https://doi.org/10.1016/j.entcs.2007.01.018

Jasmin Christian Blanchette, Lorenzo Gheri, Andrei Popescu, and Dmitriy Traytel. 2019. Bindings as bounded natural functors. *Proc. ACM Program. Lang.* 3, POPL (2019), 22:1–22:34. https://doi.org/10.1145/3290335

Luca Cardelli, Simone Martini, John C. Mitchell, and Andre Scedrov. 1994. An Extension of System F with Subtyping. *Inf. Comput.* 109, 1/2 (1994), 4–56. https://doi.org/10.1006/inco.1994.1013

Arthur Charguéraud. 2012. The Locally Nameless Representation. *J. Autom. Reason.* 49, 3 (2012), 363–408. https://doi.org/10.1007/s10817-011-9225-2

James Cheney. 2006. Completeness and Herbrand theorems for nominal logic. *J. Symb. Log.* 71, 1 (2006), 299–320. https://doi.org/10.2178/JSL/1140641176

Pierre-Louis Curien and Giorgio Ghelli. 1992. Coherence of Subsumption, Minimum Typing and Type-Checking in F<=. *Math. Struct. Comput. Sci.* 2, 1 (1992), 55–91. https://doi.org/10.1017/S0960129500001134

N. G. de Bruijn. 1972. Lambda Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church–Rosser Theorem. *Indag. Math* 75, 5 (1972), 381–392. https://doi.org/10.1016/1385-7258(72)90034-0

M. Dickmann. 1985. Larger infinitary languages. In *Handbook of Model Theoretic Logics*. Springer-Verlag, 150–171.

Gilles Dowek and Murdoch James Gabbay. 2012. Permissive-nominal logic: First-order logic over nominal terms and sets. *ACM Trans. Comput. Log.* 13, 3 (2012), 20:1–20:36. https://doi.org/10.1145/2287718.2287720

Gilles Dowek and Murdoch James Gabbay. 2023. PNL to HOL: from the logic of nominal sets to the logic of higher-order functions. *CoRR* abs/2312.16239 (2023). https://doi.org/10.48550/ARXIV.2312.16239 arXiv:2312.16239

Gilles Dowek, Murdoch James Gabbay, and Dominic P. Mulligan. 2010. Permissive nominal terms and their unification: an infinite, co-infinite approach to nominal techniques. *Log. J. IGPL* 18, 6 (2010), 769–822. https://doi.org/10.1093/JIGPAL/JZQ006

Amy P. Felty and Alberto Momigliano. 2012. Hybrid: A Definitional Two-Level Approach to Reasoning with Higher-Order Abstract Syntax. *J. Autom. Reasoning* 48, 1 (2012), 43–105. https://doi.org/10.1007/s10817-010-9194-x

Marcelo P. Fiore, Gordon D. Plotkin, and Daniele Turi. 1999. Abstract Syntax and Variable Binding. In *Logic in Computer Science (LICS) 1999*. IEEE Computer Society, 193–202. https://doi.org/10.1109/LICS.1999.782615

Murdoch Gabbay. 2007. A general mathematics of names. *Inf. Comput.* 205, 7 (2007), 982–1011. https://doi.org/10.1016/J.IC.2006.10.010

Murdoch Gabbay and Andrew M. Pitts. 1999. A New Approach to Abstract Syntax Involving Binders. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 214–224. https://doi.org/10.1109/LICS.1999.782617

Murdoch Gabbay and Andrew M. Pitts. 2002. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects Comput.* 13, 3-5 (2002), 341–363. https://doi.org/10.1007/s001650200016

Lorenzo Gheri and Andrei Popescu. 2020. A Formalized General Theory of Syntax with Bindings: Extended Version. *J. Autom. Reason.* 64, 4 (2020), 641–675. https://doi.org/10.1007/S10817-019-09522-2

M. J. C. Gordon and T. F. Melham (Eds.). 1993. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press.

William P. Hanf. 1964. Languages with Expressions of Infinite Length. *Journal of Symbolic Logic* 33, 3 (1964), 477–478. https://doi.org/10.2307/2270356

Robert Harper, Furio Honsell, and Gordon D. Plotkin. 1987. A Framework for Defining Logics. In *Logic in Computer Science (LICS) 1987*. IEEE Computer Society, 194–204. https://doi.org/10.1145/138027.138060

John Harrison. 2024. The HOL Light Theorem Prover. http://www.cl.cam.ac.uk/~jrh13/hol-light/.

Matthew Hennessy and Colin Stirling. 1985. The Power of the Future Perfect in Program Logics. *Inf. Control.* 67, 1-3 (1985), 23–52. https://doi.org/10.1016/S0019-9958(85)80025-5

Brian Huffman and Ondřej Kunčar. 2013. Lifting and Transfer: A Modular Design for Quotients in Isabelle/HOL. In *Certified Programs and Proofs*, Georges Gonthier and Michael Norrish (Eds.). Springer International Publishing, Cham, 131–146. https://doi.org/10.1007/978-3-319-03545-1_9

Felix Joachimski. 2001. *Reduction Properties of* ΠIE*-Systems*. Ph. D. Dissertation. LMU München.

Jonas Kaiser, Brigitte Pientka, and Gert Smolka. 2017. Relating System F and $\lambda 2$: A Case Study in Coq, Abella and Beluga. In *Formal Structures for Computation and Deduction (FSCD) 2017*, Dale Miller (Ed.). LIPIcs, Vol. 84. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, 21:1–21:19. https://doi.org/10.4230/LIPIcs.FSCD.2017.21

Florian Kammüller, Markus Wenzel, and Lawrence C. Paulson. 1999. Locales - A Sectioning Concept for Isabelle. In *Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99 (Lecture Notes in Computer Science, Vol. 1690)*, Yves Bertot, Gilles Dowek, André Hirschowitz, Christine Paulin-Mohring, and Laurent Théry (Eds.). Springer, 149–166. https://doi.org/10.1007/3-540-48256-3_11

H. Jerome Keisler. 1971. *Model Theory for Infinitary Logic*. North-Holland Pub. Co., Amsterdam,.

B. Knaster. 1928. Un théorème sur les fonctions d'ensembles. *Ann. Soc. Polon. Math.* 6 (1928), 133–134.

Alexander Kurz, Daniela Petrişan, Paula Severi, and Fer-Jan de Vries. 2012. An Alpha-Corecursion Principle for the Infinitary Lambda Calculus. In *Coalgebraic Methods in Computer Science (CMCS) 2012*, Dirk Pattinson and Lutz Schröder (Eds.). LNCS, Vol. 7399. Springer, 130–149. https://doi.org/10.1007/978-3-642-32784-1_8

Alexander Kurz, Daniela Petrişan, Paula Severi, and Fer-Jan de Vries. 2013. Nominal Coalgebraic Data Types with Applications to Lambda Calculus. *Logical Methods in Computer Science* 9, 4 (2013). https://doi.org/10.2168/LMCS-9(4:20)2013

Jean-Jacques Lévy. 1975. An algebraic interpretation of the lambda beta - calculus and a labeled lambda - calculus. In *Lambda-Calculus and Computer Science Theory, Proceedings of the Symposium Held in Rome, Italy, March 25-27, 1975 (Lecture Notes in Computer Science, Vol. 37)*, Corrado Böhm (Ed.). Springer, 147–165. https://doi.org/10.1007/BFb0029523

Michael Makkai. 1969. On the Model Theory of Denumerably Long Formulas with Finite Strings of Quantifiers. *J. Symb. Log.* 34, 3 (1969), 437–459. https://doi.org/10.2307/2270908

Michael Makkai and Robert Paré. 1989. *Accessible Categories: The Foundations of Categorical Model Theory*. Providence.

David Marker. 2016. *Lectures on Infinitary Model Theory*. Cambridge University Press, New York, NY, USA.

Damiano Mazza. 2012. An Infinitary Affine Lambda-Calculus Isomorphic to the Full Lambda-Calculus. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*. 471–480. https://doi.org/10.1109/LICS.2012.57

James McKinna and Robert Pollack. 1999. Some Lambda Calculus and Type Theory Formalized. *J. Autom. Reason.* 23, 3-4 (1999), 373–409.

Stefan Milius and Thorsten Wißmann. 2015. Finitary Corecursion for the Infinitary Lambda Calculus. In *6th Conference on Algebra and Coalgebra in Computer Science, CALCO 2015, June 24-26, 2015, Nijmegen, The Netherlands (LIPIcs, Vol. 35)*, Lawrence S. Moss and Pawel Sobocinski (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 336–351. https://doi.org/10.4230/LIPICS.CALCO.2015.336

R. Milner. 1989. *Communication and Concurrency*. Prentice-Hall, Inc., USA.

Robin Milner. 1993. The Polyadic π-Calculus: a Tutorial. In *Logic and Algebra of Specification*, Friedrich L. Bauer, Wilfried Brauer, and Helmut Schwichtenberg (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 203–246.

Robin Milner. 1999. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press.

Robin Milner, Joachim Parrow, and David Walker. 1992. A Calculus of Mobile Processes, I/II. *Inf. Comput.* 100, 1 (1992), 1–77. https://doi.org/10.1016/0890-5401(92)90008-4

Tobias Nipkow, Lawrence C. Paulson, and Markarius Wenzel. 2002. *Isabelle/HOL: a proof assistant for higher-order logic.* Vol. 2283. Springer Science & Business Media.

Michael Norrish. 2004. Recursive Function Definition for Types with Binders. In *Theorem Proving in Higher Order Logics (TPHOLs) 2004*, Konrad Slind, Annette Bunker, and Ganesh Gopalakrishnan (Eds.). LNCS, Vol. 3223. Springer, 241–256. https://doi.org/10.1007/978-3-540-30142-4_18

Michael Norrish. 2006. Mechanising lambda-calculus using a classical first order theory of terms with permutations. *High. Order Symb. Comput.* 19, 2-3 (2006), 169–195. https://doi.org/10.1007/S10990-006-8745-7

Michael Norrish and René Vestergaard. 2007. Proof Pearl: De Bruijn Terms Really Do Work. In *Theorem Proving in Higher Order Logics, 20th International Conference, TPHOLs 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4732)*, Klaus Schneider and Jens Brandt (Eds.). Springer, 207–222. https://doi.org/10.1007/978-3-540-74591-4_16

Frank Pfenning and Conal Elliott. 1988. Higher-Order Abstract Syntax. In *Programming Language Design and Implementation (PLDI) 1988*, Richard L. Wexelblat (Ed.). ACM, 199–208. https://doi.org/10.1145/53990.54010

Frank Pfenning and Carsten Schürmann. 1999. System Description: Twelf—A Meta-Logical Framework for Deductive Systems. In *Conference on Automated Deduction (CADE) 1999*, Harald Ganzinger (Ed.). LNCS, Vol. 1632. Springer, 202–206. https://doi.org/10.1007/3-540-48660-7_14

Brigitte Pientka. 2010. Beluga: Programming with Dependent Types, Contextual Data, and Contexts. In *Functional and Logic Programming (FLOPS) 2010*, Matthias Blume, Naoki Kobayashi, and Germán Vidal (Eds.). LNCS, Vol. 6009. Springer, 1–12. https://doi.org/10.1007/978-3-642-12251-4_1

Andrew M. Pitts. 2003. Nominal Logic, a First Order Theory of Names and Binding. *Inf. Comput.* 186, 2 (2003), 165–193. https://doi.org/10.1016/S0890-5401(03)00138-X

Andrew M. Pitts. 2006. Alpha-Structural Recursion and Induction. *J. ACM* 53, 3 (2006), 459–506. https://doi.org/10.1145/1147954.1147961

Andrew M. Pitts. 2013. *Nominal Sets: Names and Symmetry in Computer Science.* Cambridge University Press. https://doi.org/10.1017/CBO9781139084673

Randy Pollack, Masahiko Sato, and Wilmer Ricciotti. 2012. A Canonical Locally Named Representation of Binding. *J. Autom. Reason.* 49, 2 (2012), 185–207. https://doi.org/10.1007/S10817-011-9229-Y

Andrei Popescu. 2024. Nominal Recursors as Epi-Recursors. *Proc. ACM Program. Lang.* 3, POPL (2024), 22:1–22:34. https://doi.org/10.1145/3290335

John C. Reynolds. 1983. Types, Abstraction and Parametric Polymorphism.. In *IFIP Congress*. 513–523.

Davide Sangiorgi and David Walker. 2001. *The π-calculus. A theory of mobile processes.* Cambridge.

Steven Schäfer, Tobias Tebbi, and Gert Smolka. 2015. Autosubst: Reasoning with de Bruijn Terms and Parallel Substitutions. In *ITP 2015 (LNCS, Vol. 9236)*, Christian Urban and Xingyuan Zhang (Eds.). Springer, 359–374. https://doi.org/10.1007/978-3-319-22102-1_24

Masako Takahashi. 1995. Parallel Reductions in lambda-Calculus. *Inf. Comput.* 118, 1 (1995), 120–127. https://doi.org/10.1006/inco.1995.1057

Alfred Tarski. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.* 5 (1955), 285–309. https://api.semanticscholar.org/CorpusID:13651629

Christian Urban. 2008. Nominal Techniques in Isabelle/HOL. *J. Autom. Reason.* 40, 4 (2008), 327–356. https://doi.org/10.1007/S10817-008-9097-2

Christian Urban, Stefan Berghofer, and Michael Norrish. 2007. Barendregt's Variable Convention in Rule Inductions. In *Conference on Automated Deduction (CADE) 2007*, Frank Pfenning (Ed.). LNCS, Vol. 4603. Springer, 35–50. https://doi.org/10.1007/978-3-540-73595-3_4

Christian Urban and Cezary Kaliszyk. 2012. General Bindings and Alpha-Equivalence in Nominal Isabelle. *Logical Methods in Computer Science* 8, 2 (2012). https://doi.org/10.2168/LMCS-8(2:14)2012

Christian Urban and Michael Norrish. 2005. A formal treatment of the Barendregt variable convention in rule inductions. In *ACM SIGPLAN International Conference on Functional Programming, Workshop on Mechanized reasoning about languages with variable binding, MERLIN 2005*, Randy Pollack (Ed.). ACM, 25–32. https://doi.org/10.1145/1088454.1088458

Christian Urban and Christine Tasson. 2005. Nominal Techniques in Isabelle/HOL. In *CADE*. 38–53.

Jan van Brügge, James McKinna, Andrei Popescu, and Dmitriy Traytel. 2025a. Barendregt Convenes with Knaster and Tarski: Implementation and Mechanization Artifact. https://doi.org/10.5281/zenodo.14197983.

Jan van Brügge, James McKinna, Andrei Popescu, and Dmitriy Traytel. 2025b. Barendregt Convenes with Knaster and Tarski: Strong Rule Induction for Syntax with Bindings—Technical Report. https://doi.org/10.5281/zenodo.14198069.

Philip Wadler. 1989. Theorems for Free!. In *FPCA '89*. ACM, 347–359.

Received 2024-07-11; accepted 2024-11-07