# Coinductive proof search for polarized logic with applications to full intuitionistic propositional logic

**José Espírito Santo**
Centre of Mathematics, University of Minho, Portugal

**Ralph Matthes**
CNRS, Institut de Recherche en Informatique de Toulouse (IRIT), France

**Luís Pinto**
Centre of Mathematics, University of Minho, Portugal

## Abstract

The approach to proof search dubbed "coinductive proof search", and previously developed by the authors for implicational intuitionistic logic, is in this paper extended to $LJP$, a focused sequent-calculus presentation of polarized intuitionistic logic, including an array of positive and negative connectives. As before, this includes developing a coinductive description of the search space generated by a sequent, an equivalent inductive syntax describing the same space, and decision procedures for inhabitation problems in the form of predicates defined by recursion on the inductive syntax. We prove the decidability of existence of focused inhabitants, and of finiteness of the number of focused inhabitants for polarized intuitionistic logic, by means of such recursive procedures. Moreover, the polarized logic can be used as a platform from which proof search for other logics is understood. We illustrate the technique with $LJT$, a focused sequent calculus for full intuitionistic propositional logic (including disjunction). For that, we have to work out the "negative translation" of $LJT$ into $LJP$ (that sees all intuitionistic types as negative types), and verify that the translation gives a faithful representation of proof search in $LJT$ as proof search in the polarized logic. We therefore inherit decidability of both problems studied for $LJP$ and thus get new proofs of these results for $LJT$.

**2012 ACM Subject Classification** Theory of computation → Type theory; Theory of computation → Proof theory

**Keywords and phrases** Inhabitation problems, Coinduction, Lambda-calculus, Polarized logic

**Digital Object Identifier** 10.4230/LIPIcs.TYPES.2020.7

**Funding** The first and the last authors were partially financed by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia) within the Projects UIDB/00013/2020 and UIDP/00013/2020. All authors got financial support by the COST action CA15123 EUTYPES.

**Acknowledgements** We would like to thank for the careful and thoughtful review by an anonymous referee.

## 1 Introduction and Motivation

An approach to proof search dubbed "coinductive proof search" has been developed by the authors [5, 7]. The approach is based on three main ideas: (i) the Curry-Howard paradigm of representation of proofs (by typed $\lambda$-terms) is extended to solutions of proof-search problems (a solution is a run of the proof search process that, if not completed, does not fail to apply bottom-up an inference rule, so it may be an infinite object); (ii) two typed $\lambda$-calculi are developed for the effect, one being obtained by a co-inductive reading of the grammar of proof terms, the other being obtained by enriching the grammar of proof terms with a formal fixed-point operator to represent cyclic behaviour, the first calculus acting as the universe for the mathematical definition of concepts pertaining to proof search (e. g., the existence


of solutions for a given logical sequent), the second calculus acting as the finitary setting where algorithmic counterparts of those concepts can be found; (iii) formal (finite) sums are employed throughout to represent choice points, so not only solutions but even entire solution spaces are represented, both coinductively and finitarily.

The approach was developed systematically for intuitionistic implicational logic, delivering new solutions to inhabitation and counting problems, and proofs of the state-of-the-art coherence theorems, in the simply typed $\lambda$-calculus [8]; it also helped the investigation of new questions, like the various concepts of finiteness suggested by proof search [6].

The goal of this paper is to extend this approach to *polarized*, intuitionistic propositional logic with a rich choice of positive and negative connectives [17, 4], and to proof search in a full-fledged focused sequent calculus. Polarized logic can be used as a platform from which proof search for other logics is understood [15]. The extension to polarized logic also aims at obtaining results about proof search for full intuitionistic propositional logic.

In this paper, coinductive proof search is applied to $LJP$, a focused sequent-calculus presentation of polarized logic. The extension works smoothly, which is a sign of the robustness of the approach, that has been developed for a relatively simple logic. Only the luxuriant syntax (typical of focused systems, rich in various forms of judgments) puts a notational challenge, and we make a proposal for that. Unlike the case of implicational logic we described in previous work of ours, guardedness of the coinductively described expressions is not enforced by the grammar alone, and so it has to be made an extra assumption; and focusing suggests a refinement of our approach: formal sums are not needed in the inversion phases, and the infinity of solutions must go infinitely often through stable sequents (this can be expressed by a rather simple instance of the parity condition). In the end, we obtain for $LJP$ decidability of provability, and decidability of finiteness of the number of proofs, with our typical two-staged decision procedure: a function that calculates the finitary representation (in the calculus with formal fixed points) of the solution space of the given logical sequent, composed with a syntax-directed, recursive predicate that tests the desired property.

As said, from the results about the polarized logic, we can extract results for other logics. We illustrate the technique with $LJT$, a focused sequent calculus for full intuitionistic propositional logic (including disjunction) [11, 3]. For that, we define the "negative translation" of $LJT$ into $LJP$, that sees all intuitionistic formulas as negative formulas (an idea rooted in the $!A \multimap B$ translation by Girard of intuitionistic logic into linear logic, and developed in various contexts [19, 15, 1]). While the translation of formulas is mostly dictated by polarity, there are subtle problems with a definition of the translation of proof terms without knowing the logical sequent they witness (see the definitions of $\mathsf{DLV}(t)$ and atomic and positive spines in Section 5). Soundness of a translation is its first aim, but we also crucially need to guarantee that the translation gives a faithful representation of proof search in $LJT$ as proof search in $LJP$. In proving this result, we benefited from the language of proof terms developed for polarized logic in [4].

**Plan of the paper.** The sequent-calculus presentation of polarized logic from [4] is reviewed in Section 2. Coinductive proof search for $LJP$ occupies Sections 3 and 4. Applications to full intuitionistic logic are extracted in Section 5. Section 6 concludes.

## 2 Background on the system $LJP$ of polarized propositional logic

We introduce the sequent calculus $LJP$ for polarized intuitionistic propositional logic (PIPL). $LJP$ is a variant of the cut-free fragment of $\lambda^{\pm}_{G}$ [4].

*Formulas* of $LJP$ are as follows (unchanged from $\lambda^{\pm}_{G}$):

$$\begin{array}{rrcl} \text{(formulas)} & A & ::= & N \mid P \\ \text{(negative)} & N, M & ::= & C \mid a^{-} \\ \text{(composite negative)} & C & ::= & \uparrow P \mid P \supset N \mid N \wedge M \\ \text{(positive)} & P, Q & ::= & a^{+} \mid \downarrow N \mid \bot \mid P \vee Q \end{array}$$

Here, we assume a supply of (names of) atoms, denoted typically by $a$—the markers $-$ and $+$ for polarity are added to the atom (name) as superscripts, giving rise to negative resp. positive atoms. The symbols $\bot$, $\wedge$ and $\vee$ obviously stand for falsity, conjunction and disjunction, $\supset$ stands for implication, and $\uparrow$ and $\downarrow$ are polarity shifts (as they are commonly denoted in the literature). We call right formulas or R-formulas positive formulas and negative atoms. The set of formulas is thus partitioned in two ways: into negative and positive formulas, and into composite negative and right formulas. The second partitioning plays an important role in $LJP$, more than in $\lambda^{\pm}_{G}$. We also use the notion of left formulas or L-formulas: they are either negative formulas or positive atoms.

*Proof terms* of $LJP$ are organized in five syntactic categories as follows:

$$\begin{array}{lrcl} \text{(values)} & v & ::= & z \mid \mathsf{thunk}(t) \mid \mathsf{in}^{P}_{i}(v) \\ \text{(terms)} & t & ::= & \lceil e \rceil \mid \ulcorner e \urcorner \mid \lambda p \mid \langle t_1, t_2 \rangle \\ \text{(co-values/spines)} & s & ::= & \mathsf{nil} \mid \mathsf{cothunk}(p) \mid v :: s \mid i :: s \\ \text{(co-terms)} & p & ::= & z^{a^{+}}.e \mid x^{N}.e \mid \mathsf{abort}^{A} \mid [p_1, p_2] \\ \text{(stable expressions)} & e & ::= & \mathsf{dlv}(t) \mid \mathsf{ret}(v) \mid \mathsf{coret}(x, s) \end{array}$$

where $i \in \{1, 2\}$, and $z$ and $x$ range over countable sets of variables assumed to be disjoint, called positive resp. negative variables.[1] The syntax deviates from $\lambda^{\pm}_{G}$ [4, Figure 4] in the following ways: the letters to denote values and covalues are now in lower case, the two expressions to type the cut rules are absent, and the last form of values (the injections) and $\mathsf{abort}$ come with type information, as well as the binding occurrences of variables in the first two forms of co-terms—all the other syntax elements do not introduce variable bindings, in particular, there is no binding in $\lambda p$ or $\mathsf{coret}(x, s)$. Often we refer to all proof terms of $LJP$ as *expressions*, and use letter $T$ to range over expressions in this wide sense ($T$ being reminiscent of terms, but not confined to the syntactic category $t$). To shorten notation, we communicate $\langle t_1, t_2 \rangle$ and $[p_1, p_2]$ as $\langle t_i \rangle_i$ and $[p_i]_i$, respectively.

We also use the typical letters for denoting elements of the syntactic categories as sorts: let $S := \{v, t, s, p, e\}$ be their set, and use letter $\tau$ to denote any element of $S$.

Since proof terms of $LJP$ come with some extra type information as compared to $\lambda^{\pm}_{G}$, the typing rules will be adjusted accordingly. The typing relation will also be slightly reduced: it is assumed that the $Focus_L$-rule of $\lambda^{\pm}_{G}$ (the one typing the $\mathsf{coret}$ construction for proof terms) only applies if the right-hand side formula is an R-formula. This also means that *focus negative left* sequents can be restricted to R-formulas on the right-hand side, which we therefore do in $LJP$.

There are five forms of *sequents*, one for each syntatic category $\tau$ of proof terms (the full names and the rationales of the categories are found in [4]):

[1] At first sight, these proof terms are far removed from any familiar sort of $\lambda$-terms; and the fact that cut-elimination does not belong to this paper means that no reduction semantics will be given here to help grasping what they are. As detailed in [4], this language refines call-by-push-value [14], with the positive/negative distinction being related to the value/computation distinction. In Section 5 the translation of the more familiar proof terms from $LJT$ into these proof terms gives some insight. Bear in mind proof terms are the cornerstone of coinductive proof search, as both the coinductive and the finitary representations of search spaces are based on them.

**Figure 1** Inductive definition of typing rules of $LJP$

$$\frac{}{\Gamma, z : a^+ \vdash [z : a^+]} \qquad \frac{\Gamma \Longrightarrow t : N}{\Gamma \vdash [\mathsf{thunk}(t) :\downarrow N]} \qquad \frac{\Gamma \vdash [v : P_i]}{\Gamma \vdash [\mathsf{in}_i^{P_{3-i}}(v) : P_1 \vee P_2]}\ i \in \{1,2\}$$

$$\frac{\Gamma \vdash e : a^-}{\Gamma \Longrightarrow \ulcorner e \urcorner : a^-} \qquad \frac{\Gamma \vdash e : P}{\Gamma \Longrightarrow \lceil e \rceil :\uparrow P} \qquad \frac{\Gamma \mid p : P \Longrightarrow N}{\Gamma \Longrightarrow \lambda p : P \supset N} \qquad \frac{\Gamma \Longrightarrow t_i : N_i \quad \text{for } i = 1,2}{\Gamma \Longrightarrow \langle t_i \rangle_i : N_1 \wedge N_2}$$

$$\frac{}{\Gamma[\mathsf{nil} : a^-] \vdash a^-} \qquad \frac{\Gamma \mid p : P \Longrightarrow R}{\Gamma[\mathsf{cothunk}(p) :\uparrow P] \vdash R} \qquad \frac{\Gamma \vdash [v : P] \quad \Gamma[s : N] \vdash R}{\Gamma[v :: s : P \supset N] \vdash R}$$

$$\frac{\Gamma[s : N_i] \vdash R}{\Gamma[i :: s : N_1 \wedge N_2] \vdash R}\ i \in \{1,2\} \qquad \frac{\Gamma, z : a^+ \vdash e : A}{\Gamma \mid z^{a^+}.e : a^+ \Longrightarrow A} \qquad \frac{\Gamma, x : N \vdash e : A}{\Gamma \mid x^N.e :\downarrow N \Longrightarrow A}$$

$$\frac{}{\Gamma \mid \mathsf{abort}^A :\bot \Longrightarrow A} \qquad \frac{\Gamma \mid p_1 : P_1 \Longrightarrow A \quad \Gamma \mid p_2 : P_2 \Longrightarrow A}{\Gamma \mid [p_i]_i : P_1 \vee P_2 \Longrightarrow A}$$

$$\frac{\Gamma \Longrightarrow t : C}{\Gamma \vdash \mathsf{dlv}(t) : C} \qquad \frac{\Gamma \vdash [v : P]}{\Gamma \vdash \mathsf{ret}(v) : P} \qquad \frac{\Gamma, x : N[s : N] \vdash R}{\Gamma, x : N \vdash \mathsf{coret}(x, s) : R}$$

| | | | |
|---|---|---|---|
| (focus negative left) | $\Gamma[s : N] \vdash R$ | (focus positive right) | $\Gamma \vdash [v : P]$ |
| (invert positive left) | $\Gamma \mid p : P \Longrightarrow A$ | (invert negative right) | $\Gamma \Longrightarrow t : N$ |
| (stable) | $\Gamma \vdash e : A$ | | |

The rules, given in Fig. 1, are the obvious adaptations of the ones in [4, Figures 1–3] (omitting the cut rules), given the more annotated syntax and the mentioned restrictions to R-formulas in some places. We recall that $\Gamma$ is a context made of associations of variables with left formulas that respect polarity, hence these associations are either $z : a^+$ or $x : N$ (in other words, positive variables are assigned atomic types only). The extra annotations ensure uniqueness of typing in that, given the shown context, type and term information, there is at most one formula that can replace any of the placeholders in $\Gamma[s : N] \vdash \cdot$, $\Gamma \vdash [v : \cdot]$, $\Gamma \mid p : \cdot \Longrightarrow \cdot$, $\Gamma \Longrightarrow t : \cdot$ and $\Gamma \vdash e : \cdot$.

We also consider sequents without proof-term annotations, i. e., $\Gamma \vdash [P]$, $\Gamma \Longrightarrow N$, $\Gamma[N] \vdash R$, $\Gamma \mid P \Longrightarrow A$ and $\Gamma \vdash A$, that we will call logical sequents. The letters $\rho, \rho'$ etc. will range over $\Gamma \vdash R$, with an R-formula on the right-hand side. Those will be called R-stable sequents. (Such logical sequents cannot be proven by a proof term of the form $\mathsf{dlv}(t)$.) Results about all forms of sequents can sometimes be presented uniformly, with the following notational device: If $\sigma$ is any logical sequent and $T$ a proof term of the suitable syntactic category, let $\sigma(T)$ denote the sequent obtained by placing "$T :$" properly into $\sigma$, e. g., if $\sigma = (\Gamma \mid P \Longrightarrow A)$, then $\sigma(p) = (\Gamma \mid p : P \Longrightarrow A)$ (the parentheses around sequents are often used for better parsing of the text). We sometimes indicate the syntactic category $\tau$ of $T$ as upper index of $\sigma$, e. g., an arbitrary logical sequent $\Gamma \vdash A$ is indicated by $\sigma^e$.

We also use the set $S$ of sorts to give a more uniform view of the different productions of the grammar of $LJP$ proof terms. E. g., we consider $\mathsf{thunk}(\cdot)$ as a unary function symbol, which is typed/sorted as $t \to v$, to be written as $\mathsf{thunk}(\cdot) : t \to v$. As another example, we see co-pairing as binary function symbol $[\cdot, \cdot] : p, p \to p$. This notational device does not take into account variable binding, and we simply consider $z^{a^+}.\cdot$ as a unary function symbol for every

$z$ and every $a$. The positive variables $z$ have no special role either in this view, so they are all nullary function symbols (i. e., constants) with sort $v$. Likewise, for every negative variable $x$, $\mathsf{coret}(x,\cdot)$ is a unary function symbol sorted as $s \to e$. We can thus see the definition of proof terms of $LJP$ as based on an infinite signature, with function symbols $f$ of arities $k \leq 2$. The inductive definition of proof terms of $LJP$ can then be depicted in the form of one rule scheme:

$$\frac{f : \tau_1, \ldots, \tau_k \to \tau \quad T_i : \tau_i, 1 \leq i \leq k}{f(T_1, \ldots, T_k) : \tau}$$

Later we will write $f(T_i)_i$ in place of $f(T_1, \ldots, T_k)$ and assume that $k$ is somehow known. Instead of writing the $k$ hypotheses $T_i : \tau_i$, we will then just write $\forall i,\, T_i : \tau_i$.

## 3 Coinductive approach to proof search in the polarized system $LJP$

In this section, we adapt our coinductive approach to proof search from implicational intuitionistic logic to $LJP$. Due to the high number of syntactic categories and different constructors for proof terms, we use the extra notational devices from the end of Section 2 to ensure a uniform presentation of mostly similar rules that appear in definitions. Our previous development sometimes departs from such a uniformity, which is why we also widen the grammar of "forests". This in turn asks for a mathematically more detailed presentation of some coinductive proofs that are subtle but lie at the heart of our analysis. (For reasons of limited space, that presentation was moved into Appendix A.5.)

### 3.1 Search for inhabitants in $LJP$, coinductively

System $LJP^{co}_{\Sigma}$ extends the proof terms of $LJP$ in two directions: there is a coinductive reading of the rules of the grammar of proof terms, and formal sums are added to the grammar as means to express alternatives. This general idea is refined when applied to the focused system $LJP$: the coinductive reading will be attached to stable expressions only; and the formal sums are not added to the categories of (co)terms, since (co)terms serve to represent the inversion phase in proof search, where choice is not called for.

The expressions in the wide sense of $LJP^{co}_{\Sigma}$ are called *forests* and ranged by the letter $T$. They comprise five categories introduced by the simultaneous coinductive definition of the sets $v^{co}_{\Sigma}$, $t^{co}_{\Sigma}$, $s^{co}_{\Sigma}$, $p^{co}_{\Sigma}$, and $e^{co}_{\Sigma}$. However, we will continue to use the sorts $\tau$ taken from the set $S$ that was introduced for $LJP$. This allows us to maintain the function-symbol view of $LJP$ with the same symbols $f$ that keep their typing/sorting. As said, only for the classes of values, spines and expressions, we add finite sums, denoted with the multiary function symbols $\Sigma^{\tau}$ for $\tau \in \{v, s, e\}$. The definition of the set of forests, i. e., the expressions (in a wide sense) of $LJP^{co}_{\Sigma}$ can thus be expressed very concisely as being obtained by only two rule schemes:

$$\frac{f : \tau_1, \ldots, \tau_k \to \tau \quad \forall i.\, T_i : \tau_i}{f(T_1, \ldots, T_k) : \tau}\ \text{coinductive if } \tau = e \qquad \frac{\forall i.\, T_i : \tau}{\sum_i^{\tau} T_i : \tau}\ \tau \in \{v, s, e\}$$

The doubly horizontal line indicates a possibly coinductive reading. As a first step, we read all these inference rules coinductively, but in a second step restrict the obtained infinitary expressions to obey the following property: infinite branches must go infinitely often through the $e$-formation rules coming from $LJP$, i. e., those depicted as unary function symbols $f : \tau_1 \to e$ (also called the *inherited* $e$-formation rules—those for $\mathsf{dlv}(\cdot)$, $\mathsf{ret}(\cdot)$ and $\mathsf{coret}(x,\cdot)$).

This can be expressed as the *parity condition* (known from parity automata where this is the acceptance condition) based on priority 2 for any rule for those $f : \tau_1 \to e$ and priority 1

**Figure 2** Predicates exfin, nofin, finfin and inffin

$$\frac{\forall i.\,\mathsf{exfin}(T_i)}{\mathsf{exfin}(f(T_i)_i)} \qquad \frac{\mathsf{exfin}(T_j)}{\mathsf{exfin}(\sum_i T_i)} \qquad \frac{\mathsf{nofin}(T_j)}{\mathsf{nofin}(f(T_i)_i)} \qquad \frac{\forall i.\,\mathsf{nofin}(T_i)}{\mathsf{nofin}(\sum_i T_i)}$$

$$\frac{\forall i.\,\mathsf{finfin}(T_i)}{\mathsf{finfin}(f(T_i)_i)} \qquad \frac{\mathsf{nofin}(T_j)}{\mathsf{finfin}(f(T_i)_i)} \qquad \frac{\forall i.\,\mathsf{finfin}(T_i)}{\mathsf{finfin}(\sum_i T_i)} \qquad \frac{\mathsf{inffin}(T_j) \quad \forall i.\,\mathsf{exfin}(T_i)}{\mathsf{inffin}(f(T_i)_i)} \qquad \frac{\mathsf{inffin}(T_j)}{\mathsf{inffin}(\sum_i T_i)}$$

for all the others. The parity condition requires that the maximum of the priorities seen infinitely often on a path in the (forest) construction is even, hence infinite cycling through the other syntactic categories and the summing operation for $e$-expressions is subordinate to infinite cycling through the inherited $e$-formation rules. Put less technically, we allow infinite branches in the construction of forests, but infinity is not allowed to come from infinite use solely of the "auxiliary" productions (for $\tau \neq e$) or the additional sum operator for $e$, thus, in particular ruling out infinite pairing with angle brackets, infinite copairing with brackets or infinite spine composition by way of one of the :: constructors—all of which would never correspond to typable proof terms—and also ruling out infinite stacks of finite sums.

Sums $\sum_i^\tau T_i$ are required to be finite and therefore may also be denoted by $T_1 + \ldots + T_k$, leaving $\tau$ implicit. We write $\mathbb{O}$ (possibly with the upper index $\tau$ that obviously cannot be inferred from the summands) for empty sums. Sums are treated as sets of alternatives (so they are identified up to associativity, commutativity and idempotency—that incorporates $\alpha$-equivalence (this is still a $\lambda$-calculus, the presentation with function symbols $f$ is a notational device) and bisimilarity coming from the full coinductive reading in the first step of the construction).

We now define an inductive notion of membership, hence restricting the notion we had in our previous papers on implicational logic.

▶ **Definition 1** (Membership). *An LJP-expression $T$ is a* member *of a forest $T'$ when the predicate* $\mathsf{mem}(T, T')$ *holds, which is defined inductively as follows.*

$$\frac{\forall i.\,\mathsf{mem}(T_i, T_i')}{\mathsf{mem}(f(T_i)_i, f(T_i')_i)} \qquad \frac{\mathsf{mem}(T, T_j')}{\mathsf{mem}(T, T_1' + \ldots + T_k')}\ \textit{for some } j$$

The intuition of this definition is obviously that the sums expressed by $\sum_i^\tau$ represent alternatives out of which one is chosen for a concrete member.

The minimum requirement for this definition to be meaningful is that the five syntactic categories are respected: if $\mathsf{mem}(T, T')$ then $T \in \tau$ iff $T' \in \tau_\Sigma^{co}$. This property holds since we tacitly assume that the sum operators are tagged with the respective syntactic category.

For a forest $T$, we call *finite extension* of $T$, which we denote by $\mathcal{E}_{\mathrm{fin}}(T)$, the set of the (finite) members of $T$, i. e., $\mathcal{E}_{\mathrm{fin}}(T) = \{T_0 \mid \mathsf{mem}(T_0, T)\}$. Properties of special interest in this paper are: (i) $\mathsf{exfinext}(T)$, defined as: $\mathcal{E}_{\mathrm{fin}}(T)$ is nonempty; and nofinext, the complement of exfinext; and (ii) $\mathsf{finfinext}(T)$, defined as: $\mathcal{E}_{\mathrm{fin}}(T)$ is finite; and inffinext, the complement of finfinext. These predicates play an important role in Section 4.

In Fig. 2, analogously to our previous work [8], we inductively characterize exfinext and finfinext, and we coinductively characterize nofinext and inffinext. Note that the characterization of finfinext resp. inffinext depends upon the characterization of nofinext resp. exfinext. In Appendix A.1, it is shown that the characterizations in Fig. 2 are adequate, namely: exfin = exfinext, nofin = nofinext, finfin = finfinext and inffin = inffinext. As immediate

**Figure 3** Solution spaces for $LJP$

$$\mathcal{S}(\Gamma \vdash [a^+]) := \sum_{(z:a^+)\in\Gamma} z \qquad \mathcal{S}(\Gamma \vdash [\bot]) := \mathbb{O}_v$$
$$\mathcal{S}(\Gamma \vdash [\downarrow N]) := \mathsf{thunk}(\mathcal{S}(\Gamma \Longrightarrow N)) \quad \mathcal{S}(\Gamma \vdash [P_1 \vee P_2]) := \sum_{i\in\{1,2\}} \mathsf{in}_i^{P_{3-i}}(\mathcal{S}(\Gamma \vdash [P_i]))$$

$$\mathcal{S}(\Gamma \Longrightarrow a^-) := \ulcorner \mathcal{S}(\Gamma \vdash a^-) \urcorner \qquad \mathcal{S}(\Gamma \Longrightarrow P \supset N) := \lambda \mathcal{S}(\Gamma \mid P \Longrightarrow N)$$
$$\mathcal{S}(\Gamma \Longrightarrow \uparrow P) := \lceil \mathcal{S}(\Gamma \vdash P) \rceil \qquad \mathcal{S}(\Gamma \Longrightarrow N_1 \wedge N_2) := \langle \mathcal{S}(\Gamma \Longrightarrow N_i) \rangle_i$$

$$\mathcal{S}(\Gamma[a^-] \vdash R) := \text{if } R = a^- \text{ then } \mathsf{nil} \text{ else } \mathbb{O}_s$$
$$\mathcal{S}(\Gamma[P \supset N] \vdash R) := \mathcal{S}(\Gamma \vdash [P]) :: \mathcal{S}(\Gamma[N] \vdash R)$$
$$\mathcal{S}(\Gamma[\uparrow P] \vdash R) := \mathsf{cothunk}(\mathcal{S}(\Gamma \mid P \Longrightarrow R))$$
$$\mathcal{S}(\Gamma[N_1 \wedge N_2] \vdash R) := \sum_{i\in\{1,2\}} (i :: \mathcal{S}(\Gamma[N_i] \vdash R))$$

$$\mathcal{S}(\Gamma \mid a^+ \Longrightarrow A) := z^{a^+}.\mathcal{S}(\Gamma, z : a^+ \vdash A) \qquad \mathcal{S}(\Gamma \mid \bot \Longrightarrow A) := \mathsf{abort}^A$$
$$\mathcal{S}(\Gamma \mid \downarrow N \Longrightarrow A) := x^N.\mathcal{S}(\Gamma, x : N \vdash A) \quad \mathcal{S}(\Gamma \mid P_1 \vee P_2 \Longrightarrow A) := [\mathcal{S}(\Gamma \mid P_i \Longrightarrow A)]_i$$

$$\mathcal{S}(\Gamma \vdash C) := \mathsf{dlv}(\mathcal{S}(\Gamma \Longrightarrow C))$$
$$\mathcal{S}(\Gamma \vdash a^-) := \sum_{(x:N)\in\Gamma} \mathsf{coret}(x, \mathcal{S}(\Gamma[N] \vdash a^-))$$
$$\mathcal{S}(\Gamma \vdash P) := \mathsf{ret}(\mathcal{S}(\Gamma \vdash [P])) + \sum_{(x:N)\in\Gamma} \mathsf{coret}(x, \mathcal{S}(\Gamma[N] \vdash P))$$

consequences, exfin and nofin are complementary predicates, as are finfin and inffin, and additionally nofin $\subseteq$ finfin.

Now, we are heading for the infinitary representation of all inhabitants of any logical sequent $\sigma$ of $LJP$ as a forest whose members are precisely those inhabitants (to be confirmed in Prop. 4). For all the five categories of logical sequents $\sigma^\tau$, we define the associated *solution space* $\mathcal{S}(\sigma^\tau)$ as a forest, more precisely, an element of $\tau_\Sigma^{co}$, that is supposed to represent the space of solutions generated by an exhaustive and possibly non-terminating search process applied to that given logical sequent $\sigma^\tau$. This is by way of the following simultaneous coinductive definition. It is simultaneous for the five categories of logical sequents. For each category, there is an exhaustive case analysis on the formula argument.

▶ **Definition 2** (Solution spaces). *We define a forest $\mathcal{S}(\sigma^\tau) \in \tau_\Sigma^{co}$ for every logical sequent $\sigma^\tau$, by simultaneous coinduction for all the $\tau \in S$. The definition is found in Fig. 3, where in the clauses for $\mathcal{S}(\Gamma \mid a^+ \Longrightarrow A)$ resp. $\mathcal{S}(\Gamma \mid \downarrow N \Longrightarrow A)$, the variables $z$ resp. $x$ are supposed to be "fresh".*

In the mentioned clauses, since the names of bound variables are considered as immaterial, there is no choice involved in this inversion phase of proof search, as is equally the case for $\mathcal{S}(\Gamma \Longrightarrow \cdot)$—as should be expected from the deterministic way inversion rules are dealt with in a focused system like $LJP$.

▶ **Lemma 3** (Well-definedness of $\mathcal{S}(\sigma)$). *For all logical sequents $\sigma$, the definition of $\mathcal{S}(\sigma)$ indeed produces a forest.*

**Proof.** Well-definedness is not at stake concerning productivity of the definition since every corecursive call is under a constructor. As is directly seen in the definition, the syntactic categories are respected. Only the parity condition requires further thought. In Appendix A.2, we prove it by showing that all the "intermediary" corecursive calls to $\mathcal{S}(\sigma)$ in the calculation of $\mathcal{S}(\Gamma \vdash A)$—which is the only case that applies inherited $e$-formation rules—lower the "weight" of the logical sequent, until a possible further call to some $\mathcal{S}(\Gamma' \vdash A')$. ◀

The members of a solution space are exactly the inhabitants of the sequent:

▶ **Proposition 4** (Adequacy of the coinductive representation). *For each $\tau \in S$, logical sequent $\sigma^\tau$ and $T$ of category $\tau$,* $\mathsf{mem}(T, \mathcal{S}(\sigma))$ *iff $\sigma(T)$ is provable in $LJP$ (proof by induction on $T$).*

The following definition is an immediate adaptation of the corresponding definition in [8].

▶ **Definition 5** (Inessential extension of contexts and R-stable sequents).

1. $\Gamma \leq \Gamma'$ *iff* $\Gamma \subseteq \Gamma'$ *and* $|\Gamma| = |\Gamma'|$, *with* $|\Delta| := \{L \mid \exists y, (y : L) \in \Delta\}$ *for an arbitrary context $\Delta$ (where we write $y$ for an arbitrary variable). That is, $\Gamma \leq \Gamma'$ if $\Gamma'$ only has extra bindings w. r. t. $\Gamma$ that come with types that are already present in $\Gamma$.*
2. $\rho \leq \rho'$ *iff for some $\Gamma \leq \Gamma'$ and for some right-formula $R$, $\rho = (\Gamma \vdash R)$ and $\rho' = (\Gamma' \vdash R)$.*

## 3.2 Search for inhabitants in $LJP$, inductively

We are going to present a finitary version of $LJP^{co}_\Sigma$ in the form of a system $LJP^{\mathsf{gfp}}_\Sigma$ of *finitary forests* that are again generically denoted by letter $T$. We are again making extensive use of our notational device introduced in Section 2. The letter $f$ ranges over the function symbols in this specific view on $LJP$. Summation is added analogously as for $LJP^{co}_\Sigma$, and there are two more constructions for the category of expressions.

$$\frac{f : \tau_1, \ldots, \tau_k \to \tau \quad \forall i.\, T_i : \tau_i}{f(T_1, \ldots, T_k) : \tau} \qquad \frac{\forall i.\, T_i : \tau}{\sum_i^\tau T_i : \tau}\ \tau \in \{v, s, e\} \qquad \frac{}{X^\rho : e} \qquad \frac{T : e}{\mathsf{gfp}\, X^\rho.T : e}$$

where $X$ is assumed to range over a countably infinite set of *fixpoint variables* and $\rho$ ranges over R-stable sequents, as said before. The conventions regarding sums $\sum_i$ in the context of forests are also assumed for finitary forests. We stress that this is an all-inductive definition, and that w. r. t. $LJP$, the same finite summation mechanism is added as for $LJP^{co}_\Sigma$, but that the coinductive generation of stable expressions is replaced by formal fixed points whose binding and bound/free variables are associated with R-stable sequents $\rho$ whose proof theory is our main aim.

Below are some immediate adaptations of definitions in our previous paper [8]. However, they are presented in the new uniform notation. Moreover, the notion of guardedness only arises with the now wider formulation of finitary forests that allows fixed-point formation for any finitary forest of the category of stable expression.

For a finitary forest $T$, let $FPV(T)$ denote the set of freely occurring typed fixed-point variables in $T$, which can be described by structural recursion:

$$FPV(f(T_i)_i) = FPV(\textstyle\sum_i T_i) = \bigcup_i FPV(T_i) \qquad FPV(X^\rho) = \{X^\rho\}$$
$$FPV(\mathsf{gfp}\, X^\rho.T) \;=\; FPV(T) \setminus \{X^{\rho'} \mid \rho' \text{ R-stable sequent and } \rho \leq \rho'\}$$

Notice the non-standard definition that considers $X^{\rho'}$ also bound by $\mathsf{gfp}\, X^\rho$, as long as $\rho \leq \rho'$. This special view on binding necessitates to study the following restriction on finitary forests: A finitary forest is called *well-bound* if, for any of its subterms $\mathsf{gfp}\, X^\rho.T$ and any free occurrence of $X^{\rho'}$ in $T$, $\rho \leq \rho'$.

▶ **Definition 6** (Interpretation of finitary forests as forests). *For a finitary forest $T$, the interpretation $[\![T]\!]$ is a forest given by structural recursion on $T$:*

$$[\![f(T_1, \ldots, T_k)]\!] = f([\![T_1]\!], \ldots, [\![T_k]\!]) \qquad [\![X^\rho]\!] = \mathcal{S}(\rho)$$
$$[\![T_1 + \ldots + T_k]\!] = [\![T_1]\!] + \ldots + [\![T_k]\!] \qquad [\![\mathsf{gfp}\, X^\rho.T]\!] = [\![T]\!]$$

**Figure 4** All other cases of the finitary representation of solution spaces for $LJP$.

$$\mathcal{F}(\Gamma \vdash [a^+];\Xi) := \sum_{(z:a^+)\in\Gamma} z \qquad \mathcal{F}(\Gamma \vdash [\downarrow N];\Xi) := \mathsf{thunk}(\mathcal{F}(\Gamma \Longrightarrow N;\Xi))$$
$$\mathcal{F}(\Gamma \vdash [\bot];\Xi) := \mathbb{O}_v \qquad \mathcal{F}(\Gamma \vdash [P_1 \vee P_2];\Xi) := \sum_{i\in\{1,2\}} \mathsf{in}_i^{P_{3-i}}(\mathcal{F}(\Gamma \vdash [P_i];\Xi))$$

$$\mathcal{F}(\Gamma \Longrightarrow a^-;\Xi) := \ulcorner \mathcal{F}(\Gamma \vdash a^-;\Xi) \urcorner \qquad \mathcal{F}(\Gamma \Longrightarrow P \supset N;\Xi) := \lambda \mathcal{F}(\Gamma \mid P \Longrightarrow N;\Xi)$$
$$\mathcal{F}(\Gamma \Longrightarrow \uparrow P;\Xi) := \lceil \mathcal{F}(\Gamma \vdash P;\Xi) \rceil \qquad \mathcal{F}(\Gamma \Longrightarrow N_1 \wedge N_2;\Xi) := \langle \mathcal{F}(\Gamma \Longrightarrow N_i;\Xi) \rangle_i$$

$$\begin{array}{rcl}
\mathcal{F}(\Gamma[a^-] \vdash R;\Xi) & := & \text{if } R = a^- \text{ then } \mathsf{nil} \text{ else } \mathbb{O}_s \\
\mathcal{F}(\Gamma[\uparrow P] \vdash R;\Xi) & := & \mathsf{cothunk}(\mathcal{F}(\Gamma \mid P \Longrightarrow R;\Xi)) \\
\mathcal{F}(\Gamma[P \supset N] \vdash R;\Xi) & := & \mathcal{F}(\Gamma \vdash [P];\Xi) :: \mathcal{F}(\Gamma[N] \vdash R;\Xi) \\
\mathcal{F}(\Gamma[N_1 \wedge N_2] \vdash R;\Xi) & := & \sum_{i\in\{1,2\}} (i :: \mathcal{F}(\Gamma[N_i] \vdash R;\Xi))
\end{array}$$

$$\begin{array}{rcll}
\mathcal{F}(\Gamma \mid a^+ \Longrightarrow A;\Xi) & := & z^{a^+}.\mathcal{F}(\Gamma, z : a^+ \vdash A;\Xi) & (z \text{ fresh}) \\
\mathcal{F}(\Gamma \mid\downarrow N \Longrightarrow A;\Xi) & := & x^N.\mathcal{F}(\Gamma, x : N \vdash A;\Xi) & (x \text{ fresh}) \\
\mathcal{F}(\Gamma \mid P_1 \vee P_2 \Longrightarrow A;\Xi) & := & [\mathcal{F}(\Gamma \mid P_i \Longrightarrow A;\Xi)]_i & \\
\mathcal{F}(\Gamma \mid \bot \Longrightarrow A;\Xi) & := & \mathsf{abort}^A &
\end{array}$$

$$\begin{array}{rcll}
\mathcal{F}(\Gamma \vdash C;\Xi) & := & \mathsf{dlv}(\mathcal{F}(\Gamma \Longrightarrow C;\Xi)) & \\
\mathcal{F}(\Gamma \vdash a^-;\Xi) & := & \mathsf{gfp}\ Y^\rho.\ \sum_{(x:N)\in\Gamma} \mathsf{coret}(x, \mathcal{F}(\Gamma[N] \vdash a^-;\ \Xi,\ Y:\rho)) & (\rho{=}\Gamma{\vdash}a^-, Y \text{ fresh}) \\
\mathcal{F}(\Gamma \vdash P;\Xi) & := & \mathsf{gfp}\ Y^\rho.\ \ \mathsf{ret}(\mathcal{F}(\Gamma \vdash [P];\ \Xi,\ Y:\rho)) & (\rho{=}\Gamma{\vdash}a^-, Y \text{ fresh}) \\
 & & \quad + \ \sum_{(x:N)\in\Gamma} \mathsf{coret}(x, \mathcal{F}(\Gamma[N] \vdash P;\ \Xi,\ Y:\rho)) &
\end{array}$$

This definition may look too simple to handle the interpretation of bound fixed-point variables adequately, and in our previous paper [8] we called an analogous definition "simplified semantics" to stress that point. However, as in that previous paper, we can study those finitary forests for which the definition is "good enough" for our purposes of capturing solution spaces: we say a finitary forest $T$ is *proper* if for any of its subterms $T'$ of the form $\mathsf{gfp}\ X^\rho.T''$, it holds that $[\![T']\!] = \mathcal{S}(\rho)$.

To any free occurrence of an $X^\rho$ in $T$ is associated a *depth*: for this, we count the function symbols on the path from the occurrence to the root and notably do not count the binding operation of fixed-point variables and the sum operations. So, $X^\rho$ only has one occurrence of depth 0 in $X^\rho$, likewise in $\mathsf{gfp}\, Y^{\rho'}.X^\rho$.

We say a finitary forest $T$ is *guarded* if for any of its subterms $T'$ of the form $\mathsf{gfp}\ X^\rho.T''$, it holds that every free occurrence in $T''$ of a fixed-point variable $X^{\rho'}$ that is bound by this fixed-point constructor has a depth of at least 1 in $T''$.

▶ **Definition 7** (Finitary solution spaces for $LJP$). *Let $\Xi := \overrightarrow{X:\rho}$ be a vector of $m \geq 0$ declarations $(X_i : \rho_i)$ where no fixed-point variable name occurs twice. The definition of the finitary forest $\mathcal{F}(\sigma;\Xi)$ is as follows. If for some $1 \leq i \leq m$, $\rho_i =: (\Gamma_i \vdash R_i) \leq \sigma$ (i. e., $\sigma = \Gamma \vdash R_i$ and $\Gamma_i \leq \Gamma$), then $\mathcal{F}(\sigma;\Xi) = X_i^\sigma$, where $i$ is taken to be the biggest such index (notice that the produced $X_i$ will not necessarily appear with the $\rho_i$ associated to it in $\Xi$). Otherwise, $\mathcal{F}(\sigma;\Xi)$ is as displayed in Fig. 4. Then, $\mathcal{F}(\sigma)$ denotes $\mathcal{F}(\sigma;\Xi)$ with empty $\Xi$.*

Analogously to the similar result for implicational logic [7, Lemma 20], one can show that $\mathcal{F}(\sigma;\Xi)$ is well-defined (the above recursive definition terminates)—some details are given in Appendix A.4. Notice that the "if-guard" in the above definition presupposes that $\sigma$ is an R-stable sequent, hence for other forms of sequents, one necessarily has to apply the (mostly recursive) rules of Fig. 4.

**Figure 5** $\mathsf{EF}_P$ and $\mathsf{NEF}_P$ predicates

$$\frac{P(\rho)}{\mathsf{EF}_P(X^\rho)} \qquad \frac{\forall i,\ \mathsf{EF}_P(T_i)}{\mathsf{EF}_P(f^*(T_i)_i)} \qquad \frac{\mathsf{EF}_P(T_j)}{\mathsf{EF}_P(\sum_i T_i)} \qquad \frac{\neg P(\rho)}{\mathsf{NEF}_P(X^\rho)} \qquad \frac{\mathsf{NEF}_P(T_j)}{\mathsf{NEF}_P(f^*(T_i)_i)} \qquad \frac{\forall i,\ \mathsf{NEF}_P(T_i)}{\mathsf{NEF}_P(\sum_i T_i)}$$

▶ **Theorem 8** (Equivalence of representations for $LJP$)**.** *Let $\sigma$ be a logical sequent and $\Xi$ as in Def. 7. We have:*

1. *$\mathcal{F}(\sigma;\Xi)$ is guarded.*
2. *$\mathcal{F}(\sigma;\Xi)$ is well-bound and $\mathcal{F}(\sigma)$ is closed.*
3. *$\mathcal{F}(\sigma;\Xi)$ is proper.*
4. *$[\![\mathcal{F}(\sigma;\Xi)]\!] = \mathcal{S}(\sigma)$; hence the coinductive and the finitary representations are equivalent.*

**Proof.** The proof is by structural induction on $\mathcal{F}(\sigma;\Xi)$. Items 1 and 2 are proved independently (the former is an easy induction, the latter on well-boundness uses in the two cases which generate $\mathsf{gfp}$-constructions the lemma "if $X^{\rho'}$ occurs free in $\mathcal{F}(\sigma;\Xi)$, then, for some $\rho \leq \rho'$, $X : \rho \in \Xi$", also proved by structural induction on $\mathcal{F}(\sigma;\Xi)$, and from that lemma follows immediately that $\mathcal{F}(\sigma)$ is closed). As in the proof of [8, Thm. 19], item 3 uses item 4, which can be proved independently, but some effort is saved if the two items are proved simultaneously. ◀

## 4 Deciding inhabitation problems in the polarized system $LJP$

Now we adapt to $LJP$ our method [8] (until now only available for intuitionistic implication) to decide *type emptiness* (provability), and to decide *type finiteness* (only finitely many inhabitants). The presentation will look very different due to our notational device. Because of the wider notion of finitary forests that does not ensure guardedness through the grammar, some subtle technical refinements will be needed in the proofs (which will involve the Prop. 9 and are detailed in Appendix A.5). In the following, we write $f^*$ to stand for a function symbol $f$ or the prefix $\mathsf{gfp}\, X^\rho.$ of a finitary forest, the latter being seen as special unary function symbol.

### 4.1 Type emptiness

We consider complementary parameterized predicates on finitary forests $\mathsf{EF}_P(T)$ and $\mathsf{NEF}_P(T)$, where the parameter $P$ is a predicate on logical sequents. ($P = \emptyset$ will be already an important case). The definition of the two predicates $\mathsf{EF}_P$ and $\mathsf{NEF}_P$ is inductive and presented in Fig. 5, although, as in [8], it is clear that they could equivalently be given by a definition by recursion over the term structure. Thus, the predicates $\mathsf{EF}_P$ and $\mathsf{NEF}_P$ are decidable if $P$ is.

The following can be proven by routine induction on $T$ (barely more than an application of de Morgan's laws): for all $T \in LJP^{\mathsf{gfp}}_\Sigma$, $\mathsf{NEF}_P(T)$ iff $\mathsf{EF}_P(T)$ does not hold.

▶ **Proposition 9** (Finitary characterization)**.** 1. *If $P \subseteq \mathsf{exfin} \circ \mathcal{S}$ and $\mathsf{EF}_P(T)$ then $\mathsf{exfin}([\![T]\!])$.*
2. *Let $T \in LJP^{\mathsf{gfp}}_\Sigma$ be well-bound, guarded and proper. If $\mathsf{NEF}_P(T)$ and for all $X^\rho \in FPV(T)$, $\mathsf{exfin}(\mathcal{S}(\rho))$ implies $P(\rho)$, then $\mathsf{nofin}([\![T]\!])$.*
3. *For any $T \in LJP^{\mathsf{gfp}}_\Sigma$ well-bound, guarded, proper and closed, $\mathsf{EF}_\emptyset(T)$ iff $\mathsf{exfin}([\![T]\!])$.*

**Proof.** 1. is proved by induction on the predicate $\mathsf{EF}_P$ (or, equivalently, on $T$). The base case for fixpoint variables needs the proviso on $P$, and all other cases are immediate by the induction hypothesis (notice the special case for $f^*$ that is even simpler).

**Figure 6** $\mathsf{FF}_P$ and $\mathsf{NFF}_P$ predicates

$$\frac{P(\rho)}{\mathsf{FF}_P(X^\rho)} \qquad \frac{\forall i,\ \mathsf{FF}_P(T_i)}{\mathsf{FF}_P(f^*(T_i)_i)} \qquad \frac{\mathsf{NEF}_\star(T_j)}{\mathsf{FF}_P(f^*(T_i)_i)} \qquad \frac{\forall i,\ \mathsf{FF}_P(T_i)}{\mathsf{FF}_P(\sum_i T_i)}$$

$$\frac{\neg P(\rho)}{\mathsf{NFF}_P(X^\rho)} \qquad \frac{\mathsf{NFF}_P(T_j) \quad \forall i,\ \mathsf{EF}_\star(T_i)}{\mathsf{NFF}_P(f^*(T_i)_i)} \qquad \frac{\mathsf{NFF}_P(T_j)}{\mathsf{NFF}_P(\sum_i T_i)}$$

2. This needs a special notion of depth of observation for the truthfulness of nofin for forests. A more refined statement has to keep track of this observation depth in premise and conclusion, even taking into account the depth of occurrences of the bound fixed-point variables of $T$. This is presented with details in Appendix A.5.

3. For $P = \emptyset$ resp. for closed $T$, the extra condition on $P$ in part 1 resp. part 2 is trivially satisfied. We now use that exfin and nofin are complements, as are $\mathsf{NEF}_P$ and $\mathsf{EF}_P$. ◀

▶ **Theorem 10** (Deciding the existence of inhabitants in $LJP$). *A logical sequent $\sigma$ of $LJP$ is inhabited iff* $\mathsf{exfin}(\mathcal{S}(\sigma))$ *iff* $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma))$. *Hence "$\sigma$ is inhabited" is decided by deciding* $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma))$. *In other words, the inhabitation problem for $LJP$ is decided by the computable predicate* $\mathsf{EF}_\emptyset \circ \mathcal{F}$.

**Proof.** The first equivalence follows by Prop. 4 and $\mathsf{exfin} = \mathsf{exfinext}$. The second equivalence follows from Prop. 9.3, using all items of Theorem 8. Computability comes from computability of the recursive function $\mathcal{F}$ and the equivalence of the inductively defined $\mathsf{EF}_\emptyset$ with a recursive procedure over the term structure of its argument. ◀

The theorem opens the way to using Prop. 9 with $P := \mathsf{EF}_\emptyset \circ \mathcal{F}$. This is explored now, but will be needed only in the next subsection. The predicates $\mathsf{EF}_\star$ and $\mathsf{NEF}_\star$ on $LJP^{\mathsf{gfp}}_\Sigma$ are defined by $\mathsf{EF}_\star := \mathsf{EF}_P$ and $\mathsf{NEF}_\star := \mathsf{NEF}_P$ for $P := \mathsf{EF}_\emptyset \circ \mathcal{F}$ (which by Theorem 10 is equivalent to say $P := \mathsf{exfin} \circ \mathcal{S}$). We already know such $P$ is decidable, hence, also $\mathsf{EF}_\star$ and $\mathsf{NEF}_\star$ are decidable. Additionally:

▶ **Lemma 11** (Sharp finitary characterization). *For all $T \in LJP^{\mathsf{gfp}}_\Sigma$, $\mathsf{EF}_\star(T)$ iff* $\mathsf{exfin}([\![T]\!])$.

**Proof.** The direction from left to right follows immediately by Proposition 9.1. The other direction is equivalent to $\mathsf{NEF}_\star(T)$ implies $\mathsf{nofin}([\![T]\!])$, which folows by an easy induction on the predicate $\mathsf{NEF}_\star$ with the help of Theorem 10 in the base case $T = X^\sigma$. ◀

## 4.2 Type finiteness

Decision of type finiteness will be achieved by mimicking the development for deciding type emptiness, but will additionally require concepts and results from the latter. The finitary characterization of type finiteness is obtained through the complementary (parametrized) predicates $\mathsf{FF}_P$ and $\mathsf{NFF}_P$, which are defined inductively on Fig. 6 and make use of the sharp finitary characterizations of emptiness and non-emptiness ($\mathsf{NEF}_\star$ and $\mathsf{EF}_\star$). That the two predicates are indeed complementary, i.e. that $\mathsf{FF}_P(T)$ iff $\mathsf{NFF}_P(T)$ does not hold, is again proved by routine induction on $T$.

▶ **Proposition 12** (Finitary characterization). 1. *If $P \subseteq \mathsf{finfin} \circ \mathcal{S}$ and $\mathsf{FF}_P(T)$ then* $\mathsf{finfin}([\![T]\!])$.
2. *Let $T \in LJP^{\mathsf{gfp}}_\Sigma$ be well-bound, guarded and proper. If $\mathsf{NFF}_P(T)$ and for all $X^\rho \in FPV(T)$,* $\mathsf{finfin}(\mathcal{S}(\rho))$ *implies $P(\rho)$, then* $\mathsf{inffin}([\![T]\!])$.

3. *For any* $T \in LJP^{\mathsf{gfp}}_{\Sigma}$ *well-bound, guarded, proper and closed,* $\mathsf{FF}_\emptyset(T)$ *iff* $\mathsf{finfin}([\![T]\!])$.

**Proof.** Each of the items follows anologously to the corresponding item of Proposition 9. In particular: 1 follows by induction on $\mathsf{FF}_P$, and uses the fact $\mathsf{nofin} \subseteq \mathsf{finfin}$; 2 needs a special notion of depth of observation for the truthfulness of $\mathsf{inffin}$ for forests, as detailed in Appendix A.5; 3 follows then by items 1 and 2, and uses the facts $\mathsf{finfin}$ and $\mathsf{inffin}$ are complements, as are $\mathsf{FF}_P$ and $\mathsf{NFF}_P$. ◀

▶ **Theorem 13** (Deciding finiteness of inhabitants in $LJP$)**.** *A logical sequent* $\sigma$ *of* $LJP$ *has (only) finitely many inhabitants iff* $\mathsf{finfin}(\mathcal{S}(\sigma))$ *iff* $\mathsf{FF}_\emptyset(\mathcal{F}(\sigma))$*. Hence "*$\sigma$ *has (only) finitely many inhabitants" is decided by deciding* $\mathsf{FF}_\emptyset(\mathcal{F}(\sigma))$*. In other words, the type finiteness problem for* $LJP$ *is decided by the computable predicate* $\mathsf{FF}_\emptyset \circ \mathcal{F}$*.*

**Proof.** The first equivalence follows by Prop. 4 and $\mathsf{finfin} = \mathsf{finfinext}$. The second equivalence follows from Prop. 12.3, using all items of Thm. 8. Computability comes from computability of the recursive function $\mathcal{F}$, decidability of $\mathsf{NEF}_\star$, and the equivalence of the inductively defined $\mathsf{FF}_\emptyset$ with a recursive procedure over the term structure of its argument. ◀

## 5 Applications to intuitionistic propositional logic with all connectives

One of the interests of polarized logic is that it can be used to analyze other logics [15]. This is also true of $LJP$ and we illustrate it now, deriving algorithms for deciding the emptiness (provability) and the finiteness problems for $LJT$ with all connectives. Such transfer of results from $LJP$ will be immediate after the preparatory work that sets up an appropriate version of $LJT$, alongside with its embedding into $LJP$.

### 5.1 System $LJT$ of intuitionistic logic with all propositional connectives

The best known variant of the focused sequent calculus $LJT$ for IPL is the one for implication only [10]. Variants including conjunction and disjunction as well can be found in [11, 3]. We present our own variant, still denoted $LJT$. *Formulas* of $LJT$ are as follows:

$$\begin{array}{rrcl} \text{(intuitionistic formulas)} & A, B & ::= & A \supset B \mid A \wedge B \mid R \\ \text{(right intuitionistic formulas)} & R & ::= & a \mid \bot \mid A \vee B \end{array}$$

where $a$ ranges over atoms, of which an infinite supply is assumed. A *positive* intuitionistic formula, $P$, is a non-atomic right intuitionistic formula.

*Proof terms* of $LJT$ are organized in three syntactic categories as follows:

$$\begin{array}{lrcl} \text{(terms)} & t & ::= & \lambda x^A.t \mid \langle t_1, t_2\rangle \mid e \\ \text{(expressions)} & e & ::= & xs \mid \mathsf{in}^A_i(t) \\ \text{(spines)} & s & ::= & \mathsf{nil} \mid t :: s \mid i :: s \mid \mathsf{abort}^R \mid [x_1^{A_1}.e_1, x_2^{A_2}e_2] \end{array}$$

where $i \in \{1, 2\}$, and $x$ ranges over a countable set of variables. We will refer to $e_1$ and $e_2$ in the latter form of spines as *arms*. Proof terms in any category are ranged over by $T$.

There are three forms of *sequents*, $\Gamma \Longrightarrow t : A$ and $\Gamma \vdash e : R$ and $\Gamma[s : A] \vdash R$, where, as usual, $\Gamma$ is a context made of associations of variables with formulas. Therefore, a logical sequent $\sigma$ in $LJT$ may have three forms: $\Gamma \Longrightarrow A$ and $\Gamma \vdash R$ and $\Gamma[A] \vdash R$. The latter two forms require a right formula to the right of the turnstile. The full definition of the typing rules of $LJT$ is given in Fig. 7. As for $LJP$, the annotations guarantee that there is at most one formula that can replace the placeholders in $\Gamma \Longrightarrow t : \cdot$, $\Gamma \vdash e : \cdot$ and $\Gamma[s : A] \vdash \cdot$.

**Figure 7** Typing rules of $LJT$

$$\frac{\Gamma, x : A \Longrightarrow t : B}{\Gamma \Longrightarrow \lambda x^A.t : A \supset B} \qquad \frac{\Gamma \Longrightarrow t_i : A_i \quad \text{for } i = 1,2}{\Gamma \Longrightarrow \langle t_1, t_2\rangle : A_1 \wedge A_2} \qquad \frac{\Gamma \vdash e : R}{\Gamma \Longrightarrow e : R} \qquad \frac{\Gamma, x : A[s : A] \vdash R}{\Gamma, x : A \vdash xs : R}$$

$$\frac{\Gamma \Longrightarrow t : A_i}{\Gamma \vdash \mathsf{in}_i^{A_{3-i}}(t) : A_1 \vee A_2}\ i \in \{1,2\} \qquad \frac{\Gamma \Longrightarrow t : A \quad \Gamma[s : B] \vdash R}{\Gamma[t :: s : A \supset B] \vdash R} \qquad \frac{}{\Gamma[\mathsf{nil} : a] \vdash a}$$

$$\frac{}{\Gamma[\mathsf{abort}^R : \bot] \vdash R} \qquad \frac{\Gamma[s : A_i] \vdash R}{\Gamma[i :: s : A_1 \wedge A_2] \vdash R}\ i \in \{1,2\} \qquad \frac{\Gamma, x_i : A_i \Longrightarrow e_i : R \quad \text{for } i = 1,2}{\Gamma[x_1^{A_1}.e_1, x_2^{A_2}.e_2 : A_1 \vee A_2] \vdash R}$$

**Figure 8** Negative translation

$$\begin{array}{rclcrcl}
(A \supset B)^* & = & \downarrow A^* \supset B^* & & (A \vee B)^\circ & = & \downarrow A^* \vee \downarrow B^* \\
(A \wedge B)^* & = & A^* \wedge B^* & & \bot^\circ & = & \bot \\
P^* & = & \uparrow P^\circ & & a^\circ & = & a^- \\
a^* & = & a^\circ & & & &
\end{array}$$

$$\begin{array}{rclcrcl}
(\lambda x^A.t)^* & = & \lambda(x^{A^*}.\mathsf{DLV}(t^*)) & & (xs)^* & = & \mathsf{coret}(x, s^*) \\
\langle t_1, t_2\rangle^* & = & \langle t_1^*, t_2^*\rangle & & \mathsf{in}_i^A(t)^* & = & \mathsf{ret}(\mathsf{in}_i^{\downarrow A^*}(\mathsf{thunk}(t^*))) \\
e^* & = & \ulcorner e^* \urcorner, \quad \text{if } e \text{ is atomic} & & & & \\
e^* & = & \lceil e^* \rceil, \quad \text{if } e \text{ is positive} & & & &
\end{array}$$

$$\begin{array}{rclcrcl}
\mathsf{nil}^* & = & \mathsf{nil} & & (\mathsf{abort}^R)^* & = & \mathsf{cothunk}(\mathsf{abort}^{R^\circ}) \\
(t :: s)^* & = & \mathsf{thunk}(t^*) :: s^* & & [x_1^{A_1}.e_1, x_2^{A_2}.e_2]^* & = & \mathsf{cothunk}([x_1^{A_1^*}.e_1^*, x_2^{A_2^*}.e_2^*]) \\
(i :: s)^* & = & i :: s^* & & & &
\end{array}$$

The characteristic feature of the design of $LJT$ is the restriction of the type of spines to right formulas. Since the type of $\mathsf{nil}$ is atomic, spines have to be "long"; and the arms of spines cannot be lambda-abstractions nor pairs, which is enforced by restricting the arms of spines to be expressions, rather than general terms: this is the usefulness of separating the class of expressions from the class of terms. In the typing rules, the restriction to right formulas is generated at the *select rule* (the typing rule for $xs$); and the long form is forced by the identity axiom (the typing rule for $\mathsf{nil}$) because it applies to atoms only.

We could not find in the literature the restriction of cut-free LJT we consider here, but Ferrari and Fiorentini [9] consider a presentation of IPL that enforces a similar use of right formulas, in spite of being given in natural deduction format and without proof terms. It is easy to equip this natural deduction system with proof terms and map it into $LJT$: the technique is fully developed in [4] for polarized logic, but goes back to [3]. Since the just mentioned system [9] is complete for provability, so is $LJT$.

System $LJT$ can be embedded in $LJP$. We define *the negative translation* $(\cdot)^* : LJT \to LJP$ in Fig. 8, comprising a translation of formulas and a translation of proof terms.

The translation of formulas uses an auxiliary translation of right intuitionistic formulas $R$: $R^\circ$ is a right formula (and specifically, $P^\circ$ is a positive formula). An intuitionistic formula $A$ is mapped to a negative formula $A^*$, hence the name of the translation. At the level of proof terms: terms (resp. spines, expressions) are mapped to terms (resp. spines, stable

expressions). Definitions like $e^* = \ulcorner e^* \urcorner$ are meaningful if one thinks of the left $e$ as being tagged with the injection into terms. Use is made of the derived construction $\mathsf{DLV}(t)$, a stable expression of $LJP$, defined by $\mathsf{DLV}(\ulcorner e \urcorner) = e$ and $\mathsf{DLV}(t) = \mathsf{dlv}(t)$ otherwise. Its derived typing rule is that $\Gamma \vdash \mathsf{DLV}(t) : N$ follows from $\Gamma \Longrightarrow t : N$.

The translation of proof terms is defined for legal proof terms in $LJT$ only: $T$ is *legal* if every expression $e$ occurring in $T$ is either atomic or positive; an expression $xs$ is *atomic* (resp. *positive*) if $s$ is atomic (resp. positive), whereas an injection is positive; and a spine $s$ is *atomic* (resp. *positive*) if every "leaf" of $s$ is $\mathsf{nil}$ or $\mathsf{abort}^a$ (resp. an injection or $\mathsf{abort}^P$). Only when translating a legal $T$ can we apply the definition of $(\cdot)^*$ to $e$ as a term.

Formally, the inductive definition of atomic and positive spines is as follows:

- $\mathsf{nil}$ is atomic; $\mathsf{abort}^a$ is atomic; if $s$ is atomic, then $t :: s$ and $i :: s$ are atomic; if, for each $i = 1, 2$, $e_i = y_i s_i$ and $s_i$ is atomic, then $[x_1^{A_1}.e_1, x_2^{A_2}.e_2]$ is atomic.
- $\mathsf{abort}^P$ is positive; if $s$ is positive, then $t :: s$ and $i :: s$ are positive; if, for each $i = 1, 2$, $e_i = y_i s_i$ and $s_i$ is positive, or $e_i = \mathsf{in}_i^A(t)$, then $[x_1^{A_1}.e_1, x_2^{A_2}.e_2]$ is positive.

Suppose $\Gamma[s : A] \vdash R$ is derivable. If $R = a$ (resp. $R = P$) then $s$ is atomic (resp. positive). Hence any typable proof term of $LJT$ is legal. Moreover, if $\Gamma \vdash e : R$ then if $e$ is atomic, $R = a$ and if $e$ is positive, $R = P$.

The negative translation is easily seen to be injective. In order to state other properties of the translation, we define the logical $LJP$ sequent $\sigma^*$ for every logical $LJT$ sequent $\sigma$: $(\Gamma \Longrightarrow A)^* = (\Gamma^* \Longrightarrow A^*)$ and $(\Gamma \vdash R)^* = (\Gamma^* \vdash R^\circ)$ and $(\Gamma[A] \vdash R)^* = (\Gamma^*[A^*] \vdash R^\circ)$.

▶ **Proposition 14** (Soundness). *For all $T = t, e, s$ in $LJT$: if $\sigma(T)$ is derivable in $LJT$ then $\sigma^*(T^*)$ is derivable in $LJP$.*

**Proof.** By simultaneous induction on derivations for $\sigma(T)$. ◀

For the converse property (faithfulness), we need to understand better the image of the negative translation, which we will call the $*$-*fragment* of $LJP$. Consider the following subclass of formulas in $LJP$:

$$\begin{array}{rrcl} (*\text{-formulas}) & M, N & ::= & a^- \mid \uparrow P \mid \downarrow N \supset M \mid N \wedge M \\ (\text{positive } \circ\text{-formulas}) & P & ::= & \bot \mid \downarrow N \vee \downarrow M \end{array}$$

The positive $\circ$-formulas are separated because they are useful to define $\circ$-formulas $R$, which are either atoms $a^-$ or positive $\circ$-formulas $P$. A $*$-formula $N$ is a negative formula; a positive $\circ$-formula $P$ is a positive formula; a $\circ$-formula $R$ is a right formula. The negative translation, at the level of formulas, is a bijection from intuitionistic formulas to $*$-formulas, from positive intuitionistic formulas to positive $\circ$-formulas; and from right intuitionistic formulas to $\circ$-formulas. The respective inverse maps are denoted $|\cdot|$: they just erase the polarity shifts and the minus sign from atoms.

If we are interested in deriving in $LJP$ logical sequents of the form $\sigma^*$ only, then some obvious cuts can be applied to the grammar of proof terms of $LJP$, yielding the following grammar $\mathcal{G}$ of $*$-*proof terms*:

$$\begin{array}{lrcl} (*\text{-terms}) & t & ::= & \lceil e \rceil \mid \ulcorner e \urcorner \mid \lambda(x^N.e) \mid \langle t_1, t_2 \rangle \\ (*\text{-spines}) & s & ::= & \mathsf{nil} \mid \mathsf{cothunk}(\mathsf{abort}^R) \mid \mathsf{cothunk}([x_1^{N_1}.e_1, x_2^{N_2}.e_2]) \mid \mathsf{thunk}(t) :: s \mid i :: s \\ (*\text{-expressions}) & e & ::= & \mathsf{dlv}(t) \mid \mathsf{ret}(\mathsf{in}_i^P(\mathsf{thunk}(t))) \mid \mathsf{coret}(x, s) \end{array}$$

Here the type annotations range over formulas in the $*$-fragment.

A *legal* $*$-proof term is one where $\mathsf{dlv}(t)$ is only allowed as the body of a $\lambda$-abstraction. Legal expressions are generated by a restricted variant of the grammar above: $\mathsf{dlv}(t)$ is

forbidden as a $*$-expression *per se*, but, as a compensation, we introduce a second form of $\lambda$-abstraction, $\lambda(x^N.\mathsf{dlv}(t))$.

There is a *forgetful* map from legal $*$-terms (resp. legal $*$-spines, legal $*$-expressions) to terms (resp. spines, expressions) of $LJT$that essentially erases term decorations, and is given in detail in Appendix A.6. The negative translation only generates legal $*$-proof terms; and, since the negative translation is just a process of decoration, the forgetful map is left inverse to it: $|T^*| = T$.

▶ **Proposition 15** (Faithfulness). *For all $T$ in $LJP$: if $\sigma^*(T)$ is derivable in $LJP$, then $T$ is legal and $\sigma(|T|)$ is derivable in $LJT$ and $|T|^* = T$.*

**Proof.** By simultaneous induction on $T = t, s, r$ as generated by the grammar $\mathcal{G}$ above. ◀

By faithfulness and injectivity of the negative translation, the implications in Proposition 14 are in fact equivalences. Moreover:

▶ **Corollary 16** (Reduction of counting and inhabitation problems).

1. *There is a bijection between the set of those $T \in LJT$ such that $\sigma(T)$ is derivable in $LJT$ and the set of those $T' \in LJP$ such that $\sigma^*(T')$ is derivable in $LJP$.*
2. *There is $T \in LJT$ such that $\sigma(T)$ is derivable in $LJT$ iff there is $T' \in LJP$ such that $\sigma^*(T')$ is derivable in $LJP$.*

**Proof.** We prove the first item. The negative translation is the candidate for the bijection. Due to Proposition 14, it maps from the first set to the second. We already observed that the translation is injective. Proposition 15 guarantees that the translation is also surjective. The second item is an immediate consequence of the first. ◀

## 5.2 Deciding emptiness and finiteness in $LJT$

The "extraction" of the two decision procedures is immediate. Both procedures will be given by the composition of two recursive functions: first, $\mathcal{F}$ calculates the finitary representation of the full solution space; second, recursing on the structure of this representation, a predicate ($\mathsf{EF}_\emptyset$ or $\mathsf{FF}_\emptyset$) is decided.

**Emptiness.** Given $\sigma$ in $LJT$: $\sigma$ is inhabited in $LJT$ iff $\sigma^*$ is inhabited in $LJP$ (Cor. 16); iff $\mathsf{exfin}(\mathcal{S}(\sigma^*))$ (Prop. 4 and $\mathsf{exfin} = \mathsf{exfinext}$); iff $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma^*))$ (Thm. 10). The obtained algorithm is thus $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma^*))$. Recall from Subsec. 4.1 that, although predicate $\mathsf{EF}_\emptyset$ is given inductively, it can be equivalently given by recursion over the structure of finitary forests.

**Finiteness.** Given $\sigma$ in $LJT$: $\sigma$ has finitely many inhabitants in $LJT$ iff $\sigma^*$ has finitely many inhabitants in $LJP$ (Cor. 16); iff $\mathsf{finfin}(\mathcal{S}(\sigma^*))$ (Prop. 4 and $\mathsf{finfin} = \mathsf{finfinext}$); iff $\mathsf{FF}_\emptyset(\mathcal{F}(\sigma^*))$ (Thm. 13). The obtained algorithm is thus $\mathsf{FF}_\emptyset(\mathcal{F}(\sigma^*))$. Again, here, we should think of $\mathsf{FF}_\emptyset$ as given by its recursive description.

**Discussion.** Complexity issues are not (yet) a concern of "coinductive proof search". So far we privileged a conceptual approach, where the representation of the search space is separated from its analysis. This separation of concerns is reflected in the architecture of our decision procedures, given as the composition of $\mathcal{F}$ with a recursive predicate adequate for the specific problem at hand. This organization is modular, with $\mathcal{F}(\sigma^*)$ being reused, as we move our attention to a different decision problem; but it is not optimized, because knowing the particular predicate we want to compose $\mathcal{F}$ with, in general, suggests simplifications. Nevertheless, here are some comparisons with algorithms from the literature.

It is well-known that provability in full IPL is PSPACE-complete. In particular, [13] establishes a space bound $\mathcal{O}(n \log n)$ for this problem based on a *contraction-free* proof system.

Of course, this kind of efficiency cannot be expected from a naive implementation of our decision method above, as it would first fully compute through $\mathcal{F}$ the finitary representation of the solution space. An immediate optimization would be to compute with $\mathcal{F}$ lazily, and interleave it with the structural decision algorithm for $\mathsf{EF}_\emptyset$, thus avoiding the explicit construction of the solution space. We wonder if such kind of optimization leads to a decision algorithm for provability in full IPL within PSPACE. Note that, if our sole interest was decision of provability, it would be better to start from variants of $LJT$ like the systems $MJ^{Hist}$ [12] or $Nbu$ [9], which, in particular, block application of context-expanding rules if the formulas to be added are already present in the context (like in *total discharge convention*). However, neither the latter systems nor contraction-free systems give an appropriate basis to address questions related to the *full set of normal proofs/inhabitants*.

The work of [18] is the only one we are aware of that deals with a question of type finiteness for full IPL (but $\bot$ is not included). That work considers a cut-free $LJT$-presentation of IPL close to ours, but allowing more proofs, due to unrestricted RHS in its *contraction* rule (recall our version of $LJT$ imposes an atom or disjunction on the RHS when a formula from the context is selected to the "focus"). The work [18] uses graphs to represent the search space, and such graphs are guaranteed to be finite only in the case where contexts are sets, in other words, when the total discharge convention is assumed. The decision of type finiteness is then based on traversal of this finite graph structure and exhaustive checking for the absence of "cyclic proof structures". In our case, the decision comes by computing the result of the function $\mathcal{F}$, which gives the finitary forest representing the solution space, and then by deciding by a simple structural recursion the predicate $\mathsf{FF}_\emptyset$ on such a forest; but, again, one may compute with $\mathcal{F}$ lazily and interleave it with the structural decision of $\mathsf{FF}_\emptyset$. It should be noted that decision of type finiteness in [18] is part of more general algorithms that count the number of inhabitants of a type. In our case, counting of inhabitants is done by a function defined by structural recursion on finitary forests. This worked fine for the implicational fragment of $LJT$ [8], and we anticipate no major obstacles in extending the idea to full $LJT$.

## 6 Final remarks

We have shown that "coinductive proof search" extends to polarized intuitionistic logic [17, 4]: the basic result about the equivalence of the coinductive and finitary representation of solution spaces is obtained, as well as decidability of some predicates (one of which is provability) through recursive predicates defined over the finitary syntax.

In the presence of disjunction, focused proofs fail to be *canonical*—in e. g. [16] (Subsec. 1.7) it is observed that types with a unique canonical inhabitant may have infinitely many focused inhabitants. So, we stress again, our algorithms for type finiteness refer to the finiteness of the number of *focused* inhabitants (which are all the inhabitants according to the specific proof systems considered in this paper). The next challenge is to try our approach with the even more sophisticated systems [16] that capture canonical inhabitants, and for that we find it useful to deal with $LJP$ first.

But the study of $LJP$ has other uses, as a platform to study other logics. We illustrated this view with $LJT$, a focused proof system for intuitionistic logic, by means of the negative translation of $LJT$ into $LJP$. Variants of this translation were previously mentioned or sketched [19, 15], here we give a full treatment as a translation between languages of proof terms. By composing the properties of the negative translation with the results about polarized logic, we extract results about proof search in $LJT$ (including notably disjunction).

Our negative translation is reminiscent of Girard's translation of intuitionistic logic into linear logic. The latter translation may be seen as underlying other translations in the literature—see [1] for a study that involves polarized *linear* logic and even covers cut-elimination (our setting is cut-free and linearity plays no role). We also worked out a positive translation of cut-free $LJQ$ [2] into $LJP$, but have no space to show it. This opens the way to the study of inhabitation problems relative to call-by-value $\lambda$-terms, and for that, the results obtained here about $LJP$ will be reused.

In the context of intuitionistic implication, we obtained in [6] decidability of problems involving the concept of solution rather than inhabitant (including the problem of termination of proof search). As further future work, we plan to extend to $LJP$ such decidability results.

## References


1 Pierre-Louis Curien and Guillaume Munch-Maccagnoni. The duality of computation under focus. In Cristian S. Calude and Vladimiro Sassone, editors, *Proceedings of Theoretical Computer Science - 6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Brisbane, Australia, September 20-23, 2010*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 165–181. Springer, 2010.

2 Roy Dyckhoff and Stéphane Lengrand. Call-by-value lambda-calculus and LJQ. *J. Log. Comput.*, 17(6):1109–1134, 2007.

3 Roy Dyckhoff and Luís Pinto. A permutation-free sequent calculus for intuitionistic logic. Technical report, St Andrews University Computer Science Research Report CS/96, August 1996.

4 José Espírito Santo. The polarized $\lambda$-calculus. In Vivek Nigam and Mário Florido, editors, *11th Workshop on Logical and Semantic Frameworks with Applications, LSFA 2016, Porto, Portugal, January 1, 2016*, volume 332 of *Electronic Notes in Theoretical Computer Science*, pages 149–168. Elsevier, 2016. URL: `https://doi.org/10.1016/j.entcs.2017.04.010`.

5 José Espírito Santo, Ralph Matthes, and Luís Pinto. A coinductive approach to proof search. In David Baelde and Arnaud Carayol, editors, *Proceedings Workshop on Fixed Points in Computer Science, FICS 2013, Turino, Italy, September 1st, 2013*, volume 126 of *EPTCS*, pages 28–43, 2013.

6 José Espírito Santo, Ralph Matthes, and Luís Pinto. Decidability of several concepts of finiteness for simple types. *Fundam. Inform.*, 170(1-3):111–138, 2019.

7 José Espírito Santo, Ralph Matthes, and Luís Pinto. A coinductive approach to proof search through typed lambda-calculi. *CoRR*, abs/1602.04382v3, 2020. `arXiv:1602.04382v3`.

8 José Espírito Santo, Ralph Matthes, and Luís Pinto. Inhabitation in simply-typed lambda-calculus through a lambda-calculus for proof search. *Mathematical Structures in Computer Science*, 29:1092–1124, 2019. Also found at HAL through `https://hal.archives-ouvertes.fr/hal-02360678v1`. URL: `https://doi.org/10.1017/S0960129518000099`.

9 Mauro Ferrari and Camillo Fiorentini. Goal-oriented proof-search in natural deduction for intuitionistic propositional logic. *J. Autom. Reasoning*, 62(1):127–167, 2019.

10 Hugo Herbelin. A $\lambda$-calculus structure isomorphic to a Gentzen-style sequent calculus structure. In L. Pacholski and J. Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1995.

11 Hugo Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de $\lambda$-termes et comme calcul de stratégies gagnantes*. Ph.D. thesis, University Paris 7, January 1995.

12 Jacob Howe. *Proof Search Issues in Some Non-Classical Logics*. Ph.D. thesis, University of St.Andrews, available as University of St.Andrews Research Report CS/99/1, 1998.

13 Jörg Hudelmaier. An o(n log n)-space decision procedure for intuitionistic propositional logic. *J. Log. Comput.*, 3(1):63–75, 1993.

14 Paul Blain Levy. Call-by-push-value: Decomposing call-by-value and call-by-name. *High. Order Symb. Comput.*, 19(4):377–414, 2006.

15 Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logic. *Theor. Comput. Sci.*, 410:4747–4768, 2009.

16 Gabriel Scherer and Didier Rémy. Which simple types have a unique inhabitant? In Kathleen Fisher and John H. Reppy, editors, *Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming, ICFP 2015, Vancouver, BC, Canada, September 1-3, 2015*, pages 243–255. ACM, 2015.

17 Robert J. Simmons. Structural focalization. *ACM Trans. Comput. Log.*, 15(3):21:1–21:33, 2014.

18 J. B. Wells and Boris Yakobowski. Graph-based proof counting and enumeration with applications for program fragment synthesis. In *Logic Based Program Synthesis and Transformation, 14th International Symposium, LOPSTR 2004, Verona, Italy, August 26-28, 2004, Revised Selected Papers*, volume 3573 of *LNCS*, pages 262–277. Springer, 2004.

19 Noam Zeilberger. Focusing and higher-order abstract syntax. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 359–369. ACM, 2008.


# A Appendix with some more technical details

## A.1 On the characterization of predicates on forests in Section 3.1

▶ **Lemma 17.** 1. *Given a forest $T$,* $\mathsf{exfin}(T)$ *iff* $\mathsf{nofin}(T)$ *does not hold.*
2. *Given a forest $T$,* $\mathsf{finfin}(T)$ *iff* $\mathsf{inffin}(T)$ *does not hold.*

**Proof.** Both items are plain instances of the generic result in the style of De Morgan's laws that presents inductive predicates as complements of coinductive predicates, by a dualization operation on the underlying clauses. ◀

The following lemma shows that the predicates exfin and finfin correspond to the intended meaning in terms of the finite extensions. Additionally, the lemma shows that the negation of exfin resp. finfin holds exactly for the forests which have no finite members resp. for the forests which have infinitely many finite members.

▶ **Lemma 18** (Coinductive characterization)**.** *Given a forest $T$,*
1. $\mathsf{exfin}(T)$ *iff* $\mathcal{E}_{\mathrm{fin}}(T)$ *is non-empty, i. e.,* $\mathsf{exfin} = \mathsf{exfinext}$ *as sets of forests;*
2. $\mathsf{finfin}(T)$ *iff* $\mathcal{E}_{\mathrm{fin}}(T)$ *is finite, i. e.,* $\mathsf{finfin} = \mathsf{finfinext}$ *as sets of forests.*

**Proof.** Item 1 follows directly from the fact: $\mathsf{exfin}(T)$ iff $\mathsf{mem}(T_0, T)$ for some $T_0$. The left to right implication is proved by induction on exfin. (Recall exfin is a predicate on forests, but is defined inductively.) The right to left implication can be proved via the equivalent statement "for all $T_0$, $\mathsf{mem}(T_0, T)$ implies $\mathsf{exfin}(T)$", which follows by induction on $LJP$ proof terms $T_0$. For the case of membership in sums, it is necessary to decompose them (thanks to priority 1) until membership in an expression $f(T_i)_i$ is reached so that the argument for the first inductive clause of membership applies. Item 2 follows analogously, but also uses the fact $\mathsf{nofin} = \mathsf{nofinext}$, which is an immediate consequence of item 1 and Lemma 17. ◀

## A.2 On well-definedness of infinitary representation in Section 3.1

This section is dedicated to the proof of Lemma 3.

It remains to check the parity condition. As mentioned in the main text, this comes from the observation that all the "intermediary" corecursive calls to $\mathcal{S}(\sigma)$ in the calculation of $\mathcal{S}(\Gamma \vdash A)$—which is the only case that applies inherited $e$-formation rules—lower the "weight" of the logical sequent, until a possible further call to some $\mathcal{S}(\Gamma' \vdash A')$.

▶ **Definition 19** (weight). *Weight of a formula:* $w(\bot, a^+) := 0$, $w(a^-) := 1$, *and for composite formulas, add the weights of the components and add the following for the extra symbols:* $w(\downarrow, \wedge) := 0$, $w(\vee) := 1$, $w(\uparrow) := 2$, $w(\supset) := 3$. *Then* $w(N) \geq 1$ *and* $w(P) \geq 0$.

*Weight of context* $\Gamma$*: the sum of the weights of all the formulas associated with the variables.*

*Weight of logical sequent:* $w(\Gamma \vdash A) := w(\Gamma) + w(A)$, $w(\Gamma \Longrightarrow N) := w(\Gamma) + w(N) - 1 \geq 0$. $w(\Gamma \vdash [P]) := w(\Gamma) + w(P)$, $w(\Gamma | P \Longrightarrow A) := w(\Gamma) + w(P) + w(A) + 1$, $w(\Gamma[N] \vdash R) := w(\Gamma) + w(N) + w(R)$. *Then for all* $\sigma$, $w(\sigma) \geq 0$.

In preparation of Section A.4, we even show the following more general statement:

▶ **Lemma 20.** *Every direct corecursive call in the definition of* $\mathcal{S}(\sigma)$ *to some* $\mathcal{S}(\sigma')$ *for neither* $\sigma$ *nor* $\sigma'$ *R-stable sequents lowers the* weight *of the logical sequent.*

**Proof.** We have to show the following inequalities:

$w(\Gamma \vdash C) > w(\Gamma \Longrightarrow C)$ (the rule introducing $\mathsf{dlv}(\cdot)$ is easy to overlook but not needed for the proof of Lemma 3): this is why $\cdot \Longrightarrow \cdot$ has to weigh less

$w(\Gamma | a^+ \Longrightarrow A) > w(\Gamma, z : a^+ \vdash A)$: this is why $\cdot | \cdot \Longrightarrow \cdot$ has to weigh more (and variable names must not enter the weight of contexts $\Gamma$)

$w(\Gamma | \downarrow N \Longrightarrow A) > w(\Gamma, x : N \vdash A)$: $w(\downarrow) = 0$ suffices

$w(\Gamma \vdash [\downarrow N]) > w(\Gamma \Longrightarrow N)$: $w(\downarrow) = 0$ suffices

$w(\Gamma \vdash [P_1 \vee P_2]) > w(\Gamma \vdash [P_i])$: trivial since $w(\vee) > 0$

$w(\Gamma \Longrightarrow a^-) > w(\Gamma \vdash a^-)$ is not to be shown (and is wrong) since we hit the class of R-stable sequents

$w(\Gamma \Longrightarrow \uparrow P) > w(\Gamma \vdash P)$: this works since $\uparrow$ weighs more (given that $\cdot \Longrightarrow \cdot$ weighs less), but this inequation is not needed either

$w(\Gamma \Longrightarrow P \supset N) > w(\Gamma | P \Longrightarrow N)$: since both logical sequent weights are unfavourably modified, the weight of $\supset$ has to be so high

$w(\Gamma \Longrightarrow N_1 \wedge N_2) > w(\Gamma \Longrightarrow N_i)$: since $w(N_{3-i}) \geq 1$

$w(\Gamma[\uparrow P] \vdash R) > w(\Gamma | P \Longrightarrow R)$: this is why $\uparrow$ has to weigh more (given that $\cdot | \cdot \Longrightarrow \cdot$ weighs more)

$w(\Gamma[P \supset N] \vdash R) > w(\Gamma \vdash [P])$ and $> w(\Gamma[N] \vdash R)$: both are trivial since $w(\supset) > 0$

$w(\Gamma[N_1 \wedge N_2] \vdash R) > w(\Gamma[N_i] \vdash R)$: since $w(N_{3-i}) \geq 1$

$w(\Gamma | P_1 \vee P_2 \Longrightarrow A) > w(\Gamma | P_i \Longrightarrow A)$: trivial since $w(\vee) > 0$ ◀

It is clear that this lemma guarantees the parity condition for all $\mathcal{S}(\sigma)$.

## A.3 On forest transformation for inessential extensions in Section 3.1

If $\rho = (\Gamma \vdash R)$ and $\rho' = (\Gamma' \vdash R)$, then the result $[\rho'/\rho]T$ of the *decontraction operation* applied to $T$ is defined to be $[\Gamma'/\Gamma]T$, with the latter given as follows:

▶ **Definition 21** (Decontraction). *Let* $\Gamma \leq \Gamma'$. *For a forest* $T$ *of* $LJP^{co}_{\Sigma}$, *the forest* $[\Gamma'/\Gamma]T$ *of* $LJP^{co}_{\Sigma}$ *is defined by corecursion in Fig. 9, where, for* $w \in dom(\Gamma)$,

$$D_w := \{w\} \cup \{w' : (w' : \Gamma(w)) \in (\Gamma' \setminus \Gamma)\} \ .$$

**Figure 9** Corecursive equations for definition of decontraction

$$
\begin{array}{lcll}
[\Gamma'/\Gamma]f(T_1,\ldots,T_k) & = & f([\Gamma'/\Gamma]T_i,\ldots,[\Gamma'/\Gamma]T_k) & \text{for } f \text{ neither } z \text{ nor } \mathsf{coret}(x,\cdot) \\
[\Gamma'/\Gamma]\sum_i T_i & = & \sum_i [\Gamma'/\Gamma]T_i & \\
[\Gamma'/\Gamma]z & = & z & \text{if } z \notin dom(\Gamma) \\
[\Gamma'/\Gamma]z & = & \sum\limits_{z' \in D_z} z' & \text{if } z \in dom(\Gamma) \\
[\Gamma'/\Gamma]\mathsf{coret}(x,s) & = & \mathsf{coret}(x,[\Gamma'/\Gamma]s) & \text{if } x \notin dom(\Gamma) \\
[\Gamma'/\Gamma]\mathsf{coret}(x,s) & = & \sum\limits_{x' \in D_x} \mathsf{coret}(x',[\Gamma'/\Gamma]s) & \text{if } x \in dom(\Gamma)
\end{array}
$$

*In other words, the occurrences of variables (in the syntactic way they are introduced in the forests) are duplicated for all other variables of the same type that $\Gamma'$ has in addition.*

▶ **Lemma 22** (Solution spaces and decontraction)**.** *Let $\rho \leq \rho'$. Then $\mathcal{S}(\rho') = [\rho'/\rho]\mathcal{S}(\rho)$.*

**Proof.** Analogous to the proof for implicational logic [7]. Obviously, the decontraction operation for forests has to be used to define decontraction operations for all forms of logical sequents (analogously to the R-stable sequents, where only $\Gamma$ varies). Then, the coinductive proof is done simultaneously for all forms of logical sequents. ◀

## A.4 On termination of finitary representation in Section 3.2

Definition 7 contains recursive equations that are not justified by calls to the same function for "smaller" sequents, in particular not for the rules governing R-stable sequents as first argument. We mentioned that the proof of termination of an analogous function for implicational logic [7, Lemma 20] can be adapted to establish also termination of $\mathcal{F}(\sigma;\Xi)$ for any valid arguments. Here, we substantiate this claim.

The difficulty comes from the rich syntax of $LJP$, so that the "true" recursive structure of $\mathcal{F}(\rho;\Xi)$—for R-stable sequents that spawn the formal fixed points—gets hidden through intermediary recursive calls with the other forms of logical sequents. However, we will now argue that all those can be seen as plainly auxiliary since they just decrease the "weight" of the problem to be solved.

▶ **Lemma 23.** *Every direct recursive call in the definition of $\mathcal{F}(\sigma;\Xi)$ to some $\mathcal{F}(\sigma';\Xi')$ for neither $\sigma$ nor $\sigma'$ R-stable sequents lowers the* weight *of the first argument.*

**Proof.** This requires to check the very same inequations as in the proof of Lemma 20. ◀

The message of the lemma is that the proof search through all the other forms of logical sequents (including the form $\Gamma \vdash C$) is by itself terminating. Of course, this was to be expected. Otherwise, we could not have "solved" them by a recursive definition in $\mathcal{F}$ where only R-stable sequents ask to be hypothetically solved through fixed-point variables.

The present argument comes from an analysis that is deeply connected to $LJP$, it has nothing to do with an abstract approach of defining (infinitary or finitary) forests. As seen directly in the definition of $\mathcal{F}$, only by cycling finitely through the $\mathsf{dlv}(\cdot)$ construction is the context $\Gamma$ extended in the arguments $\sigma$ to $\mathcal{F}$. And the context of the last fixed-point variable in $\Xi$ grows in lockstep.

It is trivial to observe that all the formula material of the right-hand sides lies in the same subformula-closed sets (see [7]) as the left-hand sides (in other words, the logical sequents in the recursive calls are taken from the same formula material, and there is no reconstruction whatsoever).

Therefore, the previous proof for the implicational case [7, Lemma 20] can be carried over without substantial changes. What counts are recursive calls with first argument an R-stable sequent for the calculation when the first argument is an R-stable sequent. In the implicational case, these "big" steps were enforced by the grammar for finitary forests (and the logical sequents $\Gamma \vdash R$ had even only atomic $R$ there, but this change is rather irrelevant for the proof (instead of counting atoms, one has to count $R$ formulas for getting the measure, but this does not affect finiteness of it). The preparatory steps in the proof of [7, Lemma 20] are also easily adapted, where the $\Gamma$ part of the first argument to $\mathcal{F}$ takes the role of the context $\Gamma$ in that proof.

## A.5 Completing the proofs of Props. 9.2 and 12.2 with extra concepts

First we prove Prop. 9.2. For this, we need an auxiliary concept with which we can formulate a refinement of that proposition. From the refinement, we eventually get Prop. 9.2.

We give a sequence of approximations from above to the coinductive predicate $\mathsf{nofin}$ whose intersection characterizes the predicate. The index $n$ is meant to indicate to which observation depth of $T$ we can guarantee that $\mathsf{nofin}(T)$ holds. For this purpose, we do not take into account the summation operation as giving depth. We present the notion as a simultaneous inductive definition.

$$\frac{}{\mathsf{nofin}_0(T)} \qquad \frac{\mathsf{nofin}_n(T_j)}{\mathsf{nofin}_{n+1}(f(T_i)_i)} \text{ for some } j \qquad \frac{\forall i.\, \mathsf{nofin}_{n+1}(T_i)}{\mathsf{nofin}_{n+1}(\sum_i T_i)}$$

A guarantee up to observation depth 0 does not mean that the root symbol is suitable but the assertion is just void. Going through a function symbol requires extra depth. The child has to be fine up to a depth that is one less. As announced, the summation operation does not provide depth, which is why this simultaneous inductive definition cannot be seen as a definition of $\mathsf{nofin}_n$ by recursion over the index $n$.

By induction on the inductive definition, one can show that $\mathsf{nofin}_n$ is antitone in $n$, i. e., if $\mathsf{nofin}_{n+1}(T)$ then $\mathsf{nofin}_n(T)$.

▶ **Lemma 24** (Closure under decontraction of each $\mathsf{nofin}_n$). *Let $\rho \leq \rho'$ and $n \geq 0$. For all forests $T$, $\mathsf{nofin}_n(T)$ implies $\mathsf{nofin}_n([\rho'/\rho]T)$.*

**Proof.** By induction on the inductive definition—we profit from not counting sums as providing depth. ◀

▶ **Lemma 25** (Inductive characterization of absence of members). *Given a forest $T$. Then, $\mathsf{nofin}(T)$ iff $\mathsf{nofin}_n(T)$ for all $n$.*

**Proof.** From left to right, this is by induction on $n$. One decomposes (thanks to priority 1) the sums until one reaches finitely many expressions $f(T_i)_i$ to which the induction hypothesis applies. From right to left, one proves coinductively $R \subseteq \mathsf{nofin}$, for $R := \{T : \forall\, n \geq 0,\ \mathsf{nofin}_n(T)\}$. For example, in the case $T = f(T_i)_i \in R$, this amounts to showing $T_j \in R$ for some $j$. The assumption $\mathsf{nofin}_1(f(T_i)_i)$ already implies the existence of at least one $T_j$. The proof is then indirect: if for all $i$ we would have $T_i \notin R$, then, for each $i$, there would be an $n_i$ s. t. $\neg\mathsf{nofin}_{n_i}(T_i)$, and letting $m$ be the maximum of these $n_i$'s, $\neg\mathsf{nofin}_m(T_i)$ by antitonicity; hence we would have $\neg\mathsf{nofin}_{m+1}(T)$, but $T \in R$. ◀

For $T \in LJP^{\mathsf{gfp}}_{\Sigma}$, we write $\mathcal{A}_n(T)$ for the following assumption: For every free occurrence of some $X^\rho$ in $T$ (those $X^\rho$ are found in $FPV(T)$) such that $\neg P(\rho)$, there is an $n_0$ with $\mathsf{nofin}_{n_0}(\mathcal{S}(\rho))$ and $d + n_0 \geq n$ for $d$ the *depth* of the occurrence in $T$ as defined earlier, where sums and generations of fixed points do not contribute to depth.

Notice that, trivially $n' \leq n$ and $\mathcal{A}_n(T)$ imply $\mathcal{A}_{n'}(T)$.

▶ **Lemma 26** (Ramification of Proposition 9.2)**.** *Let $T \in LJP^{\mathsf{gfp}}_{\Sigma}$ be well-bound, proper and guarded and such that $\mathsf{NEF}_P(T)$ holds. Then, for all $n \geq 0$, $\mathcal{A}_n(T)$ implies $\mathsf{nofin}_n([\![T]\!])$.*

**Proof.** By induction on the predicate $\mathsf{NEF}_P$ (which can also be seen as a proof by induction on finitary forests).

Case $T = X^\rho$. Then $[\![T]\!] = \mathcal{S}(\rho)$. Assume $n \geq 0$ such that $\mathcal{A}_n(T)$. By inversion, $\neg P(\rho)$, hence, since $X^\rho \in FPV(T)$ at depth 0 in $T$, this gives $n_0 \geq n$ with $\mathsf{nofin}_{n_0}(\mathcal{S}(\rho))$. Since $\mathsf{nofin}_m$ is antitone in $m$, we also have $\mathsf{nofin}_n([\![T]\!])$.

Case $T = \mathsf{gfp}\, X^\rho.T_1$. $\mathsf{NEF}_P(T)$ comes from $\mathsf{NEF}_P(T_1)$. Let $N := [\![T]\!] = [\![T_1]\!]$. As $T$ is proper, $N = \mathcal{S}(\rho)$. We do the proof by a side induction on $n$. The case $n = 0$ is trivial. So assume $n = n' + 1$ and $\mathcal{A}_n(T)$ and that we already know that $\mathcal{A}_{n'}(T)$ implies $\mathsf{nofin}_{n'}(\mathcal{S}(\rho))$. We have to show $\mathsf{nofin}_n(\mathcal{S}(\rho))$, i. e., $\mathsf{nofin}_n([\![T_1]\!])$. We use the main induction hypothesis on $T_1$ with the same index $n$. Hence, it suffices to show $\mathcal{A}_n(T_1)$. Consider any free occurrence of some $Y^{\rho'}$ in $T_1$ such that $\neg P(\rho')$. We have to show that there is an $n_0$ with $\mathsf{nofin}_{n_0}(\mathcal{S}(\rho'))$ and $d + n_0 \geq n$ for $d$ the depth of the occurrence in $T_1$.

First sub-case: the considered occurrence is also a free occurrence in $T$. Since we disregard fixed-point constructions for depth, $d$ is also the depth in $T$. Because of $\mathcal{A}_n(T)$, we get an $n_0$ as desired.

Second sub-case: the remaining case is with $Y = X$ and, since $T$ is well-bound, $\rho \leq \rho'$. As remarked before, $\mathcal{A}_n(T)$ gives us $\mathcal{A}_{n'}(T)$. The side induction hypothesis therefore yields $\mathsf{nofin}_{n'}(\mathcal{S}(\rho))$. By closure of $\mathsf{nofin}_n$ under decontraction, we get $\mathsf{nofin}_{n'}([\rho'/\rho]\mathcal{S}(\rho))$, but that latter forest is $\mathcal{S}(\rho')$ by Lemma 22. By guardedness of $T$, this occurrence of $X^{\rho'}$ has depth $d \geq 1$ in $T_1$. Hence, $d + n' \geq 1 + n' = n$.

Case $T = f(T_1, \ldots, T_k)$ with a proper function symbol $f$. Assume $n \geq 0$ such that $\mathcal{A}_n(T)$. There is an index $j$ such that $\mathsf{NEF}_P(T)$ comes from $\mathsf{NEF}_P(T_j)$. Assume $n \geq 0$ such that $\mathcal{A}_n(T)$. We have to show that $\mathsf{nofin}_n([\![T]\!])$. This is trivial for $n = 0$. Thus, assume $n = n' + 1$. We are heading for $\mathsf{nofin}_{n'}([\![T_j]\!])$. We use the induction hypothesis on $T_j$ (even with this smaller index $n'$). Therefore, we are left to show $\mathcal{A}_{n'}(T_j)$. Consider any free occurrence of some $X^\rho$ in $T_j$ such that $\neg P(\rho)$, of depth $d$ in $T_j$. This occurrence is then also a free occurrence in $T$ of depth $d + 1$ in $T$. From $\mathcal{A}_n(T)$, we get an $n_0$ with $\mathsf{nofin}_{n_0}(\mathcal{S}(\rho))$ and $d + 1 + n_0 \geq n$, hence with $d + n_0 \geq n'$, hence $n_0$ is as required for showing $\mathcal{A}_{n'}(T_j)$.

Case $T = \sum_i T_i$. $\mathsf{NEF}_P(T)$ comes from $\mathsf{NEF}_P(T_i)$ for all $i$. Assume $n \geq 0$ such that $\mathcal{A}_n(T)$. We have to show that $\mathsf{nofin}_n([\![T]\!])$. This is trivial for $n = 0$. Thus, assume $n = n' + 1$ and fix some index $i$. We have to show $\mathsf{nofin}_n([\![T_i]\!])$. We use the induction hypothesis on $T_i$ (with the same index $n$). Therefore, we are left to show $\mathcal{A}_n(T_i)$. Consider any free occurrence of some $X^\rho$ in $T_i$ such that $\neg P(\rho)$, of depth $d$ in $T_i$. This occurrence is then also a free occurrence in $T$ of depth $d$ in $T$. From $\mathcal{A}_n(T)$, we get an $n_0$ with $\mathsf{nofin}_{n_0}(\mathcal{S}(\rho))$ and $d + n_0 \geq n$, hence $n_0$ is as required for showing $\mathcal{A}_n(T_i)$. (Of course, it is important that sums do not count for depth in finitary terms if they do not count for the index of the approximations to $\mathsf{nofin}$. Therefore, this proof case is so simple.) ◀

We return to Prop. 9.2:

**Proof.** Let $T \in LJP^{\mathsf{gfp}}_{\Sigma}$ be well-bound, proper and guarded, assume $\mathsf{NEF}_P(T)$ and that for all $X^\rho \in FPV(T)$, $\mathsf{exfin}(\mathcal{S}(\rho))$ implies $P(\rho)$. We have to show $\mathsf{nofin}([\![T]\!])$. By Lemma 25 it

suffices to show $\mathsf{nofin}_n([\![T]\!])$ for all $n$. Let $n \geq 0$. By the just proven refinement, it suffices to show $\mathcal{A}_n(T)$. Consider any free occurrence of some $X^\rho$ in $T$ such that $\neg P(\rho)$, of depth $d$ in $T$. By contraposition of the assumption on $FPV(T)$ and by the complementarity of $\mathsf{nofin}$ and $\mathsf{exfin}$, we have $\mathsf{nofin}(\mathcal{S}(\rho))$, hence by Lemma 25 $\mathsf{nofin}_n(\mathcal{S}(\rho))$, and $d + n \geq n$, as required for $\mathcal{A}_n(T)$. ◀

The whole development above can be replayed to prove Prop. 12.2. Now, the required auxiliary concept is $\mathsf{inffin}_n$, which gives a sequence of approximations to the coinductive predicate $\mathsf{inffin}$:

$$\frac{}{\mathsf{inffin}_0(T)} \qquad \frac{\mathsf{inffin}_n(T_j) \quad \forall i.\, \mathsf{exfin}(T_i)}{\mathsf{inffin}_{n+1}(f(T_i)_i)} \text{ for some } j \qquad \frac{\mathsf{inffin}_{n+1}(T_j)}{\mathsf{inffin}_{n+1}(\sum_i T_i)} \text{ for some } j$$

▶ **Lemma 27** ($\mathsf{inffin}_n$ antitonicity and closedness under decontraction). *Given a forest $T$ and $n \geq 0$,*

1. *if* $\mathsf{inffin}_{n+1}(T)$ *then* $\mathsf{inffin}_n(T)$*;*
2. *for any $\rho \leq \rho'$,* $\mathsf{inffin}_n(T)$ *implies* $\mathsf{inffin}_n([\rho'/\rho]T)$*.*

**Proof.** Both items 1 and 2 follow by induction on the inductive definition of $\mathsf{inffin}_n$, and 2 uses closedness of $\mathsf{exfin}$ under decontraction. ◀

▶ **Lemma 28** (Inductive characterization of finiteness of members). *Given a forest $T$,* $\mathsf{inffin}(T)$ *iff* $\mathsf{inffin}_n(T)$ *for all $n \geq 0$.*

**Proof.** Analogously to the proof of Lemma 25, the left to right direction follows by induction on $n$, and the right to left direction follows by proving coinductively $R \subseteq \mathsf{inffin}$, for $R := \{T : \forall\, n \geq 0,\, \mathsf{inffin}_n(T)\}$. ◀

For $T \in LJP^{\mathsf{gfp}}_\Sigma$, now $\mathcal{A}_n(T)$ will stand for the assumption: For every free occurrence of some $X^\rho$ in $T$ (those $X^\rho$ are found in $FPV(T)$) such that $\neg P(\rho)$, there is an $n_0$ with $\mathsf{inffin}_{n_0}(\mathcal{S}(\rho))$ and $d + n_0 \geq n$ for $d$ the depth of the occurrence in $T$ as defined earlier, where sums and generations of fixed points do not contribute to depth. (The only change w. r. t. the definition of $\mathcal{A}_n(T)$ above is the replacement of $\mathsf{nofin}_{n_0}$ by $\mathsf{inffin}_{n_0}$.)

▶ **Lemma 29** (Ramification of Proposition 12.2). *Let $T \in LJP^{\mathsf{gfp}}_\Sigma$ be well-bound, proper and guarded and such that $\mathsf{NFF}_P(T)$ holds. Then, for all $n \geq 0$, $\mathcal{A}_n(T)$ implies $\mathsf{inffin}_n([\![T]\!])$.*

**Proof.** By induction on the predicate $\mathsf{NFF}_P$. All cases for $T$ follow analogously to the corresponding cases of Lemma 26, with the help of Lemma 27. The case $T = f(T_i)_i$ uses additionally Lemma 11. ◀

Finally, Prop. 12.2 follows from Lemma 29 (in lockstep with the proof of Prop. 9.2 from Lemma 26) thanks to Lemma 28.

## A.6 Details on the forgetful map in Section 5

The *forgetful* map from legal $*$-terms (resp. legal $*$-spines, legal $*$-expressions) to terms (resp. spines, expressions) of $LJT$ is as follows:

$$\begin{array}{rclrcl}
|\lambda(x^N.\mathsf{dlv}(t))| &=& \lambda x^{|N|}.|t| & |\mathsf{nil}| &=& \mathsf{nil} \\
|\lambda(x^N.e)| &=& \lambda x^{|N|}.|e| & |\mathsf{cothunk}(\mathsf{abort}^R)| &=& \mathsf{abort}^{|R|} \\
|\langle t_1, t_2\rangle| &=& \langle |t_1|, |t_2|\rangle & \mathsf{cothunk}([x_1^{N_1}.e_1, x_2^{N_2}.e_2]) &=& [x_1^{|N_1|}.|e_1|, x_2^{|N_2|}.|e_2|] \\
|\ulcorner e \urcorner| &=& |e| & |\mathsf{thunk}(t) :: s| &=& |t| :: |s| \\
|\lceil e \rceil| &=& |e| & |i :: s| &=& i :: |s| \\
|\mathsf{coret}(x, s)| &=& x|s| & |\mathsf{ret}(\mathsf{in}_i^P(\mathsf{thunk}(t)))| &=& \mathsf{in}_i^{|P|}(|t|)
\end{array}$$