A Coinductive Approach to Proof Search through Typed Lambda-Calculi

José Espírito Santo, Ralph Matthes, Luís Pinto

August 1, 2016

Abstract

In reductive proof search, proofs are naturally generalized by solutions, comprising all (possibly infinite) structures generated by locally correct, bottom-up application of inference rules. We propose a rather natural extension of the Curry-Howard paradigm of representation, from proofs to solutions: to represent solutions by (possibly infinite) terms of the coinductive variant of the typed lambda-calculus that represents proofs. We take this as a starting point for a new, comprehensive approach to proof search; our case study is proof search in the sequent calculus LJT for intuitionistic implication logic. A second, finitary representation is proposed, where the lambda-calculus that represents proofs is extended with a formal greatest fixed point. In the latter system, fixed-point variables enjoy a relaxed form of binding that allows the detection of cycles through the type system. Formal sums are used in both representations to express alternatives in the search process, so that not only individual solutions but actually solution spaces are expressed. Moreover, formal sums are used in the coinductive syntax to define "co-contraction" (contraction bottom-up). A semantics is defined assigning a coinductive λ-term to each finitary term, making use of co-contraction as a semantical match to the relaxed form of binding of fixed-point variables present in the finitary system. The main result is the existence of an equivalent finitary representation for any given solution space expressed coinductively.

1 Introduction

Proof theory starts with the observation that a proof is more than just the truth value of a theorem. A valid theorem can have many proofs, and several of them can be interesting. In this paper, we somehow extend this to the limit and study all proofs of a given proposition. Of course, who studies proofs can also study any of them (or count them, if there are only finitely many possible proofs, or try to enumerate them in the countable case). But we do this study somehow simultaneously: we introduce a language to express the full "solution space" of proof search. And since we focus on the generative aspects of proof search, it would seem awkward to filter out failed proof attempts from the outset. This does not mean that we pursue impossible paths in the proof search (which would hardly make sense) but that we allow to follow infinite paths. An infinite path does not correspond to a successful proof, but it is a structure of locally correct proof steps. In other words, we use coinductive syntax to model *all* locally correct proof figures. This gives rise to a not necessarily wellfounded search tree. However, to keep the technical effort simpler, we have chosen a logic where this tree is finitely branching, namely the implicational fragment of intuitionistic propositional logic (with proof system given by the cut-free fragment of the sequent calculus LJT presented as the typed calculus $\overline{\lambda}$ [Her95]).

Lambda terms or variants of them (expressions that may have bound variables) are a natural means to express proofs (an observation that is called *the* Curry-Howard isomorphism) in implicational logic. Proof alternatives (locally, there are only finitely many of them since our logic has no quantifier that ranges over infinitely many individuals) can be formally represented by a finite sum of such solution space expressions, and it is natural to consider those sums up to equivalence

of the *set* of the alternatives. Since infinite lambda-terms are involved and since whole solution spaces are being modeled, we call these coinductive terms *Böhm forests*.

By their coinductive nature, Böhm forests are no proper syntactic objects: they can be defined by all mathematical (meta-theoretic) means and are thus not "concrete", as would be expected from syntactic elements. This freedom of definition will be demonstrated and exploited in the canonical definition (Definition 4) of Böhm forests as solutions to the task of proving a sequent (a formula A in a given context Γ). In a certain sense, nothing is gained by this representation: although one can calculate on a case-by-case basis the Böhm forest for a formula of interest and see that it is described as fixed point of a system of equations (involving auxiliary Böhm forests as solutions for the other meta-variables that appear in those equations), an arbitrary Böhm forest can only be observed to any finite depth, without ever knowing whether it is the expansion of a regular cyclic graph structure (the latter being a finite structure).

Therefore, a coinductive representation is more like a semantics, a mathematical definition; in particular, one cannot extract algorithms from an analysis based on it. For this reason, an alternative, *finitary* representation of solution spaces is desired, and we develop, for intuitionistic implication logic, one such representation in the form of a ("normal", i. e., inductive) typed lambda-calculus. Besides formal sums (to express choice in the search procedure), this calculus has fixed points, to capture cyclic structure; moreover, fixed-point variables enjoy a relaxed form of binding, since cycle structure has to be captured up to the inference rule of contraction.

Our main result is that the Böhm forests that appear as solution spaces of sequents can be interpreted as semantics of a typed term in this finitary typed lambda-calculus. For the Horn fragment (where nesting of implications to the left is disallowed), this works very smoothly without surprises ([EMP13, Theorem 15]). The full implicational case, however, needs some subtleties concerning the fixed-point variables over which the greatest fixed points are formed and about capturing redundancy that comes from the introduction of several hypotheses that suppose the same formula—hypotheses that would be identified by applications of the inference rule of contraction. The interpretation of the finite expressions in terms of Böhm forests needs, in the full case, a special operation, defined on Böhm forests, that we call *co-contraction* (contraction bottom-up). Without this operation, certain repetitive patterns in the solution spaces due to the presence of negative occurrences of implications could not be identified. With it, we obtain the finitary representation (Theorem 54).

The paper is organized as follows. Section 2 recalls the system $LJT/\overline{\lambda}$ and elaborates on proof search in this system. Section 3 develops the coinductive representation of solution spaces for $LJT/\overline{\lambda}$. Section 4 studies the operation of co-contraction. Section 5 develops the finitary calculus and the finitary representation of solution spaces. Section 6 concludes, also discussing related and future work.

This paper is a substantially revised and extended version of [EMP13][1]. Relatively to this work, the main novel aspects of this paper are:

1. an in-depth analysis of co-contraction;
2. the development of a typing system for the untyped finitary system $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$ of [EMP13];
3. the revision of the technical details leading to the main theorem of [EMP13] (Theorem 24), in light of the refinements allowed by the novel typing system.

2 Background

We start by introducing the cut-free fragment of system $\overline{\lambda}$, a sequent calculus for intuitionistic implication in [Her95]. Since we do not consider the cut rule here, the system is isomorphic to the system of simply-typed long normal forms in lambda-calculus, and the questions we are asking can be seen as extensions of the well-known inhabitation problem of simply-typed lambda-calculus that is likewise naturally restricted to β-normal terms and often further restricted to η-long terms.

[1] Note however that in this paper we do not treat separately the Horn fragment, as we do in [EMP13].

Figure 1: Typing rules of $\overline{\lambda}$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A.t : A \supset B}\ \mathit{RIntro} \qquad \frac{(x : \vec{B} \supset p) \in \Gamma \quad \forall i,\, \Gamma \vdash t_i : B_i}{\Gamma \vdash x\langle t_i\rangle_i : p}\ \mathit{LVecIntro}$$

2.1 $\overline{\lambda}$-system

Letters p, q, r are used to range over a base set of propositional variables (which we also call *atoms*). Letters A, B, C are used to range over the set of formulas (= types) built from propositional variables using the implication connective (that we write $A \supset B$) that is parenthesized to the right. Throughout the paper, we will use the fact that any implicational formula can be uniquely decomposed as $A_1 \supset A_2 \supset \cdots \supset A_k \supset p$ with $k \geq 0$, written in vectorial notation as $\vec{A} \supset p$. For example, if the vector $\vec{A}$ is empty the notation means simply p, and if $\vec{A} = A_1, A_2$, the notation means $A_1 \supset (A_2 \supset p)$.

A cut-free term of $\overline{\lambda}$ is either a typed lambda-abstraction or a variable applied to a possibly empty list of terms. For succinctness, instead of writing lists as a second syntactic category, we will use the informal notation $\langle t_1, \ldots, t_k\rangle$ (meaning $\langle\rangle$ if $k = 0$), abbreviated $\langle t_i\rangle_i$ if there is no ambiguity on the range of indices. So, cut-free $\overline{\lambda}$-terms are given by the following grammar:

$$\text{(terms)} \qquad t, u \quad ::= \quad \lambda x^A.t \mid x\,\langle t_1, \ldots, t_k\rangle$$

where a countably infinite set of variables ranged over by letters w, x, y, z is assumed. Note that in λ-abstractions we adopt a *domain-full* presentation (a. k. a. Church-style syntax), annotating the bound variable with a formula. As is common-place with lambda-calculi, we will throughout identify terms up to α-equivalence, i. e., names of bound variables may be consistently changed, and this is not considered as changing the term. The term constructor $x\,\langle t_1, \ldots, t_k\rangle$ is usually called *application*. When $n = 0$ we simply write the variable x. The terms are obviously in one-to-one correspondence with β-normal "ordinary" lambda-terms, the only difference being the explicit tupling of argument terms to variables in the $\overline{\lambda}$ syntax.

We will view contexts Γ as finite sets of declarations $x : A$, where no variable x occurs twice. The context $\Gamma, x : A$ is obtained from Γ by adding the declaration $x : A$, and will only be written if x is not declared in Γ. Context union is written as concatenation Γ, Δ for contexts Γ and Δ if $\Gamma \cap \Delta = \emptyset$. The letters Γ, Δ, Θ are used to range over contexts, and the notation $\mathit{dom}(\Gamma)$ stands for the set of variables declared in Γ. We will write $\Gamma(x)$ for the type associated with x for $x \in \mathit{dom}(\Gamma)$, hence viewing Γ as a function on $\mathit{dom}(\Gamma)$. Context inclusion $\Gamma \subseteq \Delta$ is just set inclusion.

In this presentation of $\overline{\lambda}$ there is only one form of sequent, namely $\Gamma \vdash t : A$. We call a sequent *atomic* when A is an atom. The rules of $\overline{\lambda}$ for deriving sequents are in Fig. 1. *LVecIntro* presupposes that the indices for the t_i range over $1, \ldots, k$ and that $\vec{B} = B_1, \ldots, B_k$, for some $k \geq 0$. Such obvious constraints for finite vectors will not be spelt out in the rest of the paper.

In the particular case of $k = 0$, in which $(x : p) \in \Gamma$ is the only hypothesis of *LVecIntro*, we type variables (with atoms). In fact, viewed in terms of the original presentation of $\overline{\lambda}$ [Her95], *LVecIntro* is a derived rule, combining logical steps of *contraction*, *left implication*, and *axiom*.

Note that the conclusion of the *LVecIntro* rule is an atomic sequent. This is not the case in [Her95], where list sequents can have a non-atomic formula on the RHS. In the variant of cut-free $\overline{\lambda}$ we adopted, the only rule available for deriving an implication is *RIntro*. Still, our atomic restriction will not cause loss of completeness of the system for intuitionistic implication. This restriction is typically adopted in systems tailored for proof search, as for example systems of focused proofs. In fact, our presentation of $\overline{\lambda}$ corresponds to a focused backward chaining system where all atoms are *asynchronous* (see e. g. [LM09]).

2.2 Reductive proof search for $\overline{\lambda}$

We consider proof search problems given by a context Γ and an implicational formula A. We express them as *sequents* $\Gamma \Rightarrow A$, corresponding to term sequents of $\overline{\lambda}$ without proof terms. $\Gamma \Rightarrow A$ is nothing but the pair consisting of Γ and A, but which is viewed as a problem description: to search for proofs of formula A in context Γ. We use the letter σ to communicate sequents in this precise sense of problem descriptions.

Even though the system $\overline{\lambda}$ is a focused sequent calculus, reductive proof search on $\overline{\lambda}$ has well identified points where choices are needed [DP99]. This is readily seen in such a simple setting as ours, where only implication is considered. Observing the rules in Fig. 1, one concludes that implications have to be decomposed by *RIntro* until an atom is obtained; here, in order to apply *LVecIntro*, a choice has to be made as to which assumption x is to be picked from the context, generating a control branching of the process (if there is no x to choose, we mark the choice point with failure); at each choice, several search sub-problems are triggered, one for each B_i, generating a different branching of the process, more of a conjunctive nature.[2] In all, a *search forest* is generated, which is *pruned* to a tree, once a choice is made at each choice point. Such trees we call *solutions* (of the proof-search problem posed by the given sequent). Sequents with solutions are called *solvable*. Since the search forest is a structure where all solutions are superimposed, we also call it *solution space*.

Finite solutions are exactly the proofs in $\overline{\lambda}$ (hence the provable sequents are solvable); but solutions need not be finite. For instance, given the sequent $\sigma = (f : p \supset p, x : p \Rightarrow p)$, we can apply forever the *LVecIntro* rule with variable f if we wish, producing an infinite solution. But σ also has finite solutions, hence is provable. On the other hand, the solvable sequent $f : p \supset p \Rightarrow p$ has a unique infinite solution, hence is not provable.

Example 1 *The illustrating examples of this paper are with the following types.*

- $\mathsf{BOOLE} := p \supset p \supset p$*, an encoding of the Boolean values as* $\lambda x^p.\lambda y^p.x$ *and* $\lambda x^p.\lambda y^p.y$*. This example illustrates that we obtain different solutions when using the differently labeled (with* x *and with* y*) hypotheses for* p*. We do not apply the so-called total discharge convention and stay plainly in the spirit of lambda-calculus.*

- $\mathsf{INFTY} := (p \supset p) \supset p$*, which is obviously uninhabited in lambda-calculus (as would be the type* p *alone), but, as mentioned before, has a unique infinite solution (see Ex. 2).*

- $\mathsf{CHURCH} := (p \supset p) \supset p \supset p$*, the type of Church numerals* $\lambda f^{p \supset p}.\lambda x^p.f^n\langle x\rangle$*,* $n \geq 0$*. As mentioned above, there is also the solution with an infinite repetition of* f*'s.*

- $\mathsf{PEIRCE} := ((p \supset q) \supset p) \supset p$ *with different atoms* p *and* q *(the Peirce formula, in particular when reading* q *as falsity), which is a classical tautology but not one of minimal logic and therefore uninhabited in lambda-calculus.*

- $\mathsf{DNPEIRCE} := (\mathsf{PEIRCE} \supset q) \supset q$*, which is provable in minimal logic and therefore inhabited in lambda-calculus (already studied in [EMP13]).*

- $\mathsf{THREE} := ((p \supset p) \supset p) \supset p$*, the simplest type of rank 3 (the nesting depth) which has inhabitants of the form* $\lambda x.x\langle\lambda y_1.x\langle\lambda y_2.x\langle\cdots\langle\lambda y_n.y_i\rangle\cdots\rangle\rangle\rangle$*,* $n \geq 1$ *and* $1 \leq i \leq n$*. (The types* $(p \supset p) \supset p$ *of* x *and* p *of all* y_k *have been omitted for presentation purposes.) Notice that* THREE *is* PEIRCE *with identification of the two atoms. It may be seen as a simplification of the* DNPEIRCE *example.*

Some of our examples are also covered in Sect. 1.3.8 of [BDS13]. Notice that they write BOOLE as 1_2 (their example (i)), CHURCH as $1 \to 0 \to 0$ (their example (iv)) and THREE as 3 (their example (vii)) in that book. PEIRCE is their example (iii).

[2] Of course, this is all too reminiscent of or- and and-branching in logic programming. But we are not confined to the Horn fragment.

The type $\mathsf{THREE} \supset p \supset p$ is example (viii) in Sect. 1.3.8 of the cited book, and is called the "monster". Since THREE is PEIRCE with identification of the two atoms p, q, the monster type is similarly resembling DNPEIRCE, but of rank 4 (while the latter has rank 5). For us, both types are equally challenging, insofar as both require an infinite supply of bound variables for enumerating their (normal) inhabitants, which is why we did not include the monster type in our sample of examples.

3 Coinductive representation of proof search

In this section we develop a coinductive representation of solutions and of solution spaces. This representation combines two ideas: the coinductive reading of the syntax of proofs, and the adoption of formal sums (in the case of solution spaces). Formal sums allow the definition of the operation of co-contraction, which will play a crucial role in the relationship to the finitary representation of solution spaces to be developed in the next section.

3.1 Representation of solutions: the $\overline{\lambda}^{co}$-system

We introduce now $\overline{\lambda}^{co}$, a coinductive extension of $\overline{\lambda}$. Its expressions, to be called *Böhm trees*, are formed without any consideration of well-typedness and will be the raw syntax that underlie possibly non-wellfounded proofs, i. e. solutions.

The raw syntax of these Böhm trees is presented as follows

$$N ::=_{co} \lambda x^A.N \mid x\langle N_1, \ldots, N_k\rangle \ ,$$

yielding the (co)terms of system $\overline{\lambda}^{co}$ (read coinductively, as indicated by the index *co*)—still with finite tuples $\langle N_i\rangle_i$.

Since the raw syntax is interpreted coinductively, also the typing rules have to be interpreted coinductively, which is symbolized by the double horizontal line in Fig. 2, a notation that we learnt from [NUB11]. (Of course, the formulas/types stay inductive.). This defines when $\Gamma \vdash N : A$ holds for a *finite* context Γ, a Böhm tree N and a type A, and the only difference to the rules in Fig. 1 is their coinductive reading and their reference to coinductively defined terms. When $\Gamma \vdash N : A$ holds, we say N is a solution of σ, when $\sigma = \Gamma \Rightarrow A$.

Since Böhm trees are not built in finitary ways from finitary syntax, the notion of equality is not just syntactic equality. Besides incorporating the identification of terms that only differ in the naming of their bound variables ("modulo α-equivalence"), we consider as equal terms that *finitely decompose* in the same way, which is to say that their successive deconstruction (not taking into account consistent differences in names of bound variables) according to the grammar must proceed the same way, and this to arbitrary depth. Thus, the natural notion of equality that we are using is bisimilarity modulo α-equivalence. Following mathematical practice, this is still written as plain equality (in type theory, it would have to be distinguished from definitional equality / convertibility and from propositional equality / Leibniz equality and would be a coinductive binary relation).

Example 2 *Consider* $it^\infty := \lambda f^{p\supset p}.N$ *with* $N = f\langle N\rangle$ *(this term* N *exists as an infinitely repeated application of* f*). Using coinduction on the typing relation, we can easily show* $\vdash it^\infty :$ INFTY*, and hence find a (co)inhabitant of a formula that does not correspond to a theorem in most logics.*

As expected, the restriction of the typing relation to the finite $\overline{\lambda}$-terms coincides with the typing relation of the $\overline{\lambda}$ system:

Lemma 3 *For any* $t \in \overline{\lambda}$, $\Gamma \vdash t : A$ *in* $\overline{\lambda}$ *iff* $\Gamma \vdash t : A$ *in* $\overline{\lambda}^{co}$.

Proof By induction on t, and using inversion of typing in $\overline{\lambda}$. □

Figure 2: Typing rules of $\overline{\lambda}^{co}$

$$\frac{\Gamma, x : A \vdash N : B}{\Gamma \vdash \lambda x^A.N : A \supset B}\ \mathit{RIntro}_{co} \qquad \frac{(x : \vec{B} \supset p) \in \Gamma \quad \forall i,\, \Gamma \vdash N_i : B_i}{\Gamma \vdash x\langle N_i\rangle_i : p}\ \mathit{LVecIntro}_{co}$$

Figure 3: Extra typing rule of $\overline{\lambda}^{co}_{\Sigma}$ w. r. t. $\overline{\lambda}^{co}$

$$\frac{\forall i,\, \Gamma \vdash E_i : p}{\Gamma \vdash \sum_i E_i : p}\ \mathit{Alts}$$

The idea of reading the syntax of lambda calculi coinductively is not new, see for example [Joa04] with a de Bruijn-style representation (meant to rule out potential problems with infinitely many or even all variables that occur freely in a term, problems that are immaterial for our study of terms in a finite typing context). For us, system $\overline{\lambda}^{co}$ is just a concise means of defining what solutions are in reductive proof search. However, we now move to original material.

3.2 Representation of solution spaces: the $\overline{\lambda}^{co}_{\Sigma}$ system

We now come to the coinductive representation of whole search spaces in $\overline{\lambda}$.

The set of coinductive cut-free $\overline{\lambda}$-terms with finite numbers of elimination alternatives is denoted by $\overline{\lambda}^{co}_{\Sigma}$ and is given by the following grammar:

$$\begin{array}{lrcl} \text{(terms)} & N & ::=_{co} & \lambda x^A.N \mid E_1 + \cdots + E_n \\ \text{(elim. alternatives)} & E & ::=_{co} & x\langle N_1, \ldots, N_k\rangle \end{array}$$

where both $n, k \geq 0$ are arbitrary. The terms of $\overline{\lambda}^{co}_{\Sigma}$ are also called *Böhm forests*. If we do not want to specify the syntactic category (terms or elimination alternatives), we consider them just as expressions and generically name them T, to reflect their nature as terms in a wide sense.

Note that summands cannot be lambda-abstractions.[3] We will often use $\sum_i E_i$ instead of $E_1 + \cdots + E_n$—in generic situations or if the dependency of E_i on i is clear, as well as the number of elements. If $n = 0$, we write $\mathbb{O}$ for $E_1 + \cdots + E_n$. If $n = 1$, we write E_1 for $E_1 + \cdots + E_n$ (in particular this injects the category of elimination alternatives into the category of (co)terms) and do as if $+$ was a binary operation on (co)terms. However, this will always have a unique reading in terms of our raw syntax of $\overline{\lambda}^{co}_{\Sigma}$. In particular, this reading makes $+$ associative and $\mathbb{O}$ its neutral element.

The coinductive typing rules of $\overline{\lambda}^{co}_{\Sigma}$ are the ones of $\overline{\lambda}^{co}$, together with the rule given in Fig. 3, where the sequents for coterms and elimination alternatives are not distinguished notationally.

Notice that $\Gamma \vdash \mathbb{O} : p$ for all Γ and p.

Since, like Böhm trees, Böhm forests are not built in finitary ways from finitary syntax (although the number of elimination alternatives is always finite, as is the number of elements of the tuples), their most natural notion of equality is again bisimilarity modulo α-equivalence. However, in Böhm forests, we even want to neglect the precise order of the summands and their (finite) multiplicity. We thus consider the sums of elimination alternatives as if they were sets of alternatives, i. e., we further assume that $+$ is symmetric and idempotent. This means, in particular, that this identification is used recursively when considering bisimilarity (anyway recursively modulo α-equivalence). This approach is convenient for a mathematical treatment but would be less so for a formalization on a computer: It has been shown by Picard and the second author [PM12] that bisimulation up to permutations in unbounded lists of children can be managed in a coinductive

[3] The division into two syntactic categories also forbids the generation of an infinite sum (for which $n = 2$ would suffice had the categories for N and E been amalgamated).

type even with the interactive proof assistant Coq, but it did not seem feasible to abstract away from the number of occurrences of an alternative (which is the meaning of idempotence of + in presence of symmetry), where multiplicity depends on the very same notion of equivalence that is undecidable in general.

As for $\overline{\lambda}^{co}$, we just use mathematical equality for this notion of bisimilarity on expressions of $\overline{\lambda}^{co}_{\Sigma}$, and so the sums of elimination alternatives can plainly be treated as if they were finite sets of elimination alternatives (given by finitely many elimination alternatives of which several might be identified through bisimilarity).

Definition 4 (Solution spaces) *The function $\mathcal{S}$, which takes a sequent $\sigma = (\Gamma \Rightarrow A)$ and produces a Böhm forest which is a coinductive representation of the sequent's solution space, is given corecursively as follows: In the case of an implication,*

$$\mathcal{S}(\Gamma \Rightarrow A \supset B) := \lambda x^A.\mathcal{S}(\Gamma, x : A \Rightarrow B) \ .$$

In the case of an atom p, for the definition of $\mathcal{S}(\Gamma \Rightarrow p)$, let $y_i : A_i$ be the i-th declaration in some enumeration of Γ with A_i of the form $\vec{B}_i \supset p$. Let $\vec{B}_i = B_{i,1}, \ldots, B_{i,k_i}$. Define $N_{i,j} := \mathcal{S}(\Gamma \Rightarrow B_{i,j})$. Then, $E_i := y_i\langle N_{i,j}\rangle_j$, and finally,

$$\mathcal{S}(\Gamma \Rightarrow p) := \sum_i E_i \ .$$

This is more sloppily written as

$$\mathcal{S}(\Gamma \Rightarrow p) := \sum_{(y:\vec{B}\supset p)\in\Gamma} y\langle \mathcal{S}(\Gamma \Rightarrow B_j)\rangle_j \ .$$

In this manner, we can even write the whole definition in one line:

$$\mathcal{S}(\Gamma \Rightarrow \vec{A} \supset p) := \lambda \vec{x} : \vec{A}. \sum_{(y:\vec{B}\supset p)\in\Delta} y\langle \mathcal{S}(\Delta \Rightarrow B_j)\rangle_j \tag{1}$$

with $\Delta := \Gamma, \vec{x} : \vec{A}$. The usual convention on bound variables ensures that (x's are fresh enough so that) Δ is a context.

A crucial element (for the succinctness of this definition and the rather structure-oriented further analysis) is that *RIntro* is the only way to prove an implication, hence that the leading λ-abstractions are inevitable. Then, the extended (finite) context Δ is traversed to pick variables y with formulas of the form $\vec{B} \supset p$, thus with the right atom p in the conclusion. And this spawns tuples of search spaces, for all the B_j, again w. r. t. the extended context Δ. Notice that this is a well-formed definition: for every sequent σ, $\mathcal{S}(\sigma)$ is a Böhm forest, regardless of the result of proof search for the given sequent σ, and this Böhm forest has the type prescribed by σ:

Lemma 5 *Given Γ and A, the typing $\Gamma \vdash \mathcal{S}(\Gamma \Rightarrow A) : A$ holds in $\overline{\lambda}^{co}_{\Sigma}$.*

In particular, all free variables of $\mathcal{S}(\Gamma \Rightarrow A)$ are declared in Γ.

Let us illustrate the function $\mathcal{S}$ at work with some examples.

Example 6 *One sees immediately that $\mathcal{S}(\Rightarrow \mathsf{BOOLE}) = \lambda x^p.\lambda y^p.x + y$.*

Example 7 *Observe that $\mathcal{S}(\Rightarrow \mathsf{INFTY}) = it^{\infty}$ (applying our notational conventions, and reflecting the fact that there is a unique alternative at each sum). In other words, it^{∞} solves the same equation as is prescribed for $\mathcal{S}(\Rightarrow \mathsf{INFTY})$, and so it is* the *solution (modulo =).*

Example 8 *Consider the sequent* $\Rightarrow \mathsf{CHURCH}$. *We have:*

$$\mathit{Church} := \mathcal{S}(\Rightarrow \mathsf{CHURCH}) = \lambda f^{p\supset p}.\lambda x^p.\mathcal{S}(f : p \supset p, x : p \Rightarrow p)$$

Now, observe that $\mathcal{S}(f : p \supset p, x : p \Rightarrow p) = f\langle\mathcal{S}(f : p \supset p, x : p \Rightarrow p)\rangle + x$ *is asked for. We identify* $\mathcal{S}(f : p \supset p, x : p \Rightarrow p)$ *as the solution for* N *of the equation* $N = f\langle N\rangle + x$. *Using* ν *as means to communicate solutions of fixed-point equations on the* meta-level, *we have*

$$\mathcal{S}(\Rightarrow \mathsf{CHURCH}) = \lambda f^{p\supset p}.\lambda x^p.\nu\, N.f\langle N\rangle + x$$

By unfolding *of the fixed point and by making a* choice *at each of the elimination alternatives, we can* collect *from this coterm as the finitary solutions of the sequent all the Church numerals* ($\lambda f^{p\supset p}.\lambda x^p.f^n\langle x\rangle$ *with* $n \in \mathbb{N}_0$), *together with the infinitary solution* $\lambda f^{p\supset p}.\lambda x^p.\nu\, N.f\langle N\rangle$ *(corresponding to always making the* f*-choice at the elimination alternatives).*

Example 9 *We consider now an example without nested implications (in the Horn fragment). Let* $\Gamma = x : p \supset q \supset p, y : q \supset p \supset q, z : p$, *with* $p \neq q$. *Note that the solution spaces of* p *and* q *relative to this sequent are mutually dependent and they give rise to the following system of equations:*

$$\begin{array}{lcl} N_p & = & x\langle N_p, N_q\rangle + z \\ N_q & = & y\langle N_q, N_p\rangle \end{array}$$

and so we have

$$\begin{array}{lcl} \mathcal{S}(\Gamma \Rightarrow p) & = & \nu\, N_p.x\langle N_p, \nu\, N_q.y\langle N_q, N_p\rangle\rangle + z \\ \mathcal{S}(\Gamma \Rightarrow q) & = & \nu\, N_q.y\langle N_q, \nu\, N_p.x\langle N_p, N_q\rangle + z\rangle \end{array}$$

Whereas for p *we can collect one finite solution* (z), *for* q *we can only collect infinite solutions.*

Example 10 *Let us consider* DNPEIRCE *of Ex. 1. When* q *is viewed as absurdity,* PEIRCE *is Peirce's law, and thus* DNPEIRCE *can be viewed as double negation of Peirce's law. We have the following (where in sequents we omit formulas on the LHS)*

$$\begin{array}{lcl} N_0 & = & \mathcal{S}(\Rightarrow \mathsf{DNPEIRCE}) = \lambda x^{\mathsf{PEIRCE}\supset q}.N_1 \\ N_1 & = & \mathcal{S}(x \Rightarrow q) = x\langle N_2\rangle \\ N_2 & = & \mathcal{S}\big(x \Rightarrow \mathsf{PEIRCE}\big) = \lambda y^{(p\supset q)\supset p}.N_3 \\ N_3 & = & \mathcal{S}(x, y \Rightarrow p) = y\langle N_4\rangle \\ N_4 & = & \mathcal{S}(x, y \Rightarrow p \supset q) = \lambda z^p.N_5 \\ N_5 & = & \mathcal{S}(x, y, z \Rightarrow q) = x\langle N_6\rangle \\ N_6 & = & \mathcal{S}\big(x, y, z \Rightarrow \mathsf{PEIRCE}\big) = \lambda y_1^{(p\supset q)\supset p}.N_7 \\ N_7 & = & \mathcal{S}(x, y, z, y_1 \Rightarrow p) = y\langle N_8\rangle + z + y_1\langle N_8\rangle \\ N_8 & = & \mathcal{S}(x, y, z, y_1 \Rightarrow p \supset q) = \lambda z_1^p.N_9 \\ N_9 & = & \mathcal{S}(x, y, z, y_1, z_1 \Rightarrow q) \end{array}$$

Now, in N_9 *observe that* y, y_1 *both have type* $(p \supset q) \supset p$ *and* z, z_1 *both have type* p, *and we are back at* N_5 *but with the duplicates* y_1 *of* y *and* z_1 *of* z. *Later, we will call this duplication phenomenon* co-contraction, *and we will give a finitary description of* N_0 *and, more generally, of all* $\mathcal{S}(\sigma)$ *(again, see Theorem 54). Of course, by taking the middle alternative in* N_7, *we obtain a finite proof, showing that* DNPEIRCE *is provable in* $\overline{\lambda}$.

Example 11 *For completeness, we describe the beginning of the calculations for* THREE *(for* PEIRCE *see Ex. 14).* $\mathcal{S}(\Rightarrow \mathsf{THREE}) = \lambda x^{(p\supset p)\supset p}.x\langle\lambda y^p.N\rangle$, *abbreviating* N *for* $\mathcal{S}(x : (p \supset p) \supset p, y : p \Rightarrow p)$. *Then,* $N = x\langle\lambda z^p.N'\rangle + y$, *with* $N' = \mathcal{S}(x : (p \supset p) \supset p, y : p, z : p \Rightarrow p)$. *We could further unravel the definition and provide a description of* $\mathcal{S}(\Rightarrow \mathsf{THREE})$ *up to any finite depth, but we prefer a more symbolic solution in Sect. 5 which exploits co-contraction in the same way as for the preceding example.*

Figure 4: Membership relations

$$\frac{\mathsf{mem}(M,N)}{\mathsf{mem}(\lambda x^A.M, \lambda x^A.N)} \qquad \frac{\forall i,\ \mathsf{mem}(M_i,N_i)}{\mathsf{mem}(x\langle M_i\rangle_i, x\langle N_i\rangle_i)} \qquad \frac{\mathsf{mem}(M,E_j)}{\mathsf{mem}(M,\sum_i E_i)}$$

We give a membership semantics for expressions of $\overline{\lambda}_\Sigma^{co}$ in terms of sets of terms in $\overline{\lambda}^{co}$. More precisely, the *membership relations* $\mathsf{mem}(M,N)$ and $\mathsf{mem}(M,E)$ are contained in $\overline{\lambda}^{co} \times \overline{\lambda}_\Sigma^{co}$ and $\overline{\lambda}^{co} \times E\overline{\lambda}_\Sigma^{co}$ respectively (where $E\overline{\lambda}_\Sigma^{co}$ stands for the set of elimination alternatives of $\overline{\lambda}_\Sigma^{co}$) and are given coinductively by the rules in Fig. 4.

Böhm trees have the types of the forests they are members of.

Lemma 12 (Typing of members) *For $N \in \overline{\lambda}^{co}$, $T \in \overline{\lambda}_\Sigma^{co}$, if $\Gamma \vdash T : A$ in $\overline{\lambda}_\Sigma^{co}$ and $\mathsf{mem}(N,T)$ then $\Gamma \vdash N : A$ in $\overline{\lambda}^{co}$.*

Proof It suffices to show for $N \in \overline{\lambda}^{co}$, $N' \in \overline{\lambda}_\Sigma^{co}$, if $\Gamma \vdash N' : A$ in $\overline{\lambda}_\Sigma^{co}$ and $\mathsf{mem}(N,N')$ then $\Gamma \vdash N : A$ in $\overline{\lambda}^{co}$ (replacing expression T by term N'), since from this follows easily the result for elimination alternatives (replacing T by $E \in \overline{\lambda}_\Sigma^{co}$). Let

$$R := \{(\Gamma, N, A) \mid \exists N' \in \overline{\lambda}_\Sigma^{co} \cdot \mathsf{mem}(N,N') \wedge \Gamma \vdash N' : A\}$$

By coinduction, to prove that this relation is contained in the typing relation of $\overline{\lambda}^{co}$, it suffices to show that it is *closed backward* relatively to the rules defining that typing relation—which means, roughly speaking, that for each element of R there is a typing rule which produces such element from premisses in R. More precisely, we need to show that for any $(\Gamma, N, A) \in R$, one of the following holds:

1. $A = A_0 \supset A_1$, $N = \lambda x^{A_0}.N_1$, and $(\Gamma, x : A_0\ ,\ N_1\ ,\ A_1) \in R$;
2. $A = p$, and there is $y : \vec{B} \supset p \in \Gamma$ so that $N = y\langle N_i\rangle_i$, and, for all i, $(\Gamma, N_i, B_i) \in R$.

Let $(\Gamma, N, A) \in R$. Then $\mathsf{mem}(N,N')$ and $\Gamma \vdash N' : A$, for some $N' \in \overline{\lambda}_\Sigma^{co}$. The proof proceeds by case analysis on A.

Case $A = A_0 \supset A_1$. By definition of the typing relation, we must have $N' = \lambda x^{A_0}.N_1'$ and $\Gamma, x : A_0 \vdash N_1' : A_1$, for some N_1'; and by definition of mem, we must have $N = \lambda x^{A_0}.N_1$, and $\mathsf{mem}(N_1, N_1')$, for some N_1; therefore, $(\Gamma, x : A_0\ ,\ N_1\ ,\ A_1) \in R$, by definition of R.

Case $A = p$. By definition of the typing relation, we have $N' = \sum_j E_j$ and $\Gamma \vdash E_j : p$, for all j. Then, by definition of mem, we must have, $\mathsf{mem}(N, E_j)$, for some j. Let $E_j = y\langle N_i'\rangle_i$. Again by definition of mem, $N = y\langle N_i\rangle_i$, with $\mathsf{mem}(N_i, N_i')$ for all i. Since $\Gamma \vdash y\langle N_i'\rangle_i : p$, we must have, again by definition of the typing relation, $y : \vec{B} \supset p \in \Gamma$ and $\Gamma \vdash N_i' : B_i$ for all i. Hence, for all i, $(\Gamma\ ,\ N_i\ ,\ B_i) \in R$, by definition of R. □

Now, we prove that in fact, for any search problem $\sigma = \Gamma \Rightarrow A$, the members of $\mathcal{S}(\sigma)$ are exactly the solutions of σ.

Proposition 13 *1. For $N \in \overline{\lambda}^{co}$, $\mathsf{mem}(N, \mathcal{S}(\Gamma \Rightarrow A))$ iff $\Gamma \vdash N : A$ in $\overline{\lambda}^{co}$.*

2. For $t \in \overline{\lambda}$, $\mathsf{mem}(t, \mathcal{S}(\Gamma \Rightarrow A))$ iff $\Gamma \vdash t : A$ in $\overline{\lambda}$.

Proof

We prove the first statement in detail as a further example of coinductive reasoning, the second statement follows immediately from the first by virtue of Lemma 3.

"If". Consider the relations

$$R_1 := \{(N, \mathcal{S}(\Gamma \Rightarrow A)) \mid \Gamma \vdash N : A\}$$
$$R_2 := \{(x\langle N_i\rangle_i, x\langle \mathcal{S}(\Gamma \Rightarrow B_i)\rangle_i) \mid (x : B_1, \cdots, B_k \supset p) \in \Gamma \wedge \Gamma \vdash x\langle N_1, \ldots, N_k\rangle : p\}$$

It suffices to show that $R_1 \subseteq \mathsf{mem}$, but this cannot be proven alone since mem is defined simultaneously for co-terms and elimination alternatives. We also prove $R_2 \subseteq \mathsf{mem}$, and to prove both by coinduction on the membership relations, it suffices to show that the relations R_1, R_2 are closed backward relatively to the rules defining the membership predicate, that is:

1. for any $(M, N) \in R_1$, one of the following holds:
   (a) $(M, N) = (\lambda x^A.M', \lambda x^A.N')$, and $(M', N') \in R_1$;
   (b) $N = \sum_i E_i$, and for some i, $(M, E_i) \in R_2$;
2. for any $(M, E) \in R_2$, $M = x\langle M_i\rangle_i$, and $E = x\langle N_i\rangle_i$, and for all i, $(M_i, N_i) \in R_1$

1. Take an arbitrary element of R_1, i.e. take $(M, \mathcal{S}(\Gamma \Rightarrow A))$ s.t. $\Gamma \vdash M : A$. One of the following happens:

i) $A = A_0 \supset A_1$, $M = \lambda x^{A_0}.M'$, and $\Gamma, x : A_0 \vdash M' : A_1$;

ii) $A = p$, and there is $y : \vec{B} \supset p \in \Gamma$ so that $M = y\langle M'_i\rangle_i$, and, for all i, $\Gamma \vdash M'_i : B_i$.

Case i). Note that $\mathcal{S}(\Gamma \Rightarrow A) = \lambda x^{A_0}.\mathcal{S}(\Gamma, x : A_0 \Rightarrow A_1)$. So, we need to show $(M', \mathcal{S}(\Gamma, x : A_0 \Rightarrow A_1)) \in R_1$, which follows from $\Gamma, x : A_0 \vdash M' : A_1$.

Case ii). Note that $\mathcal{S}(\Gamma \Rightarrow A) = \sum_{z:\vec{C}\supset p\in\Gamma} z\langle \mathcal{S}(\Gamma \Rightarrow C_j)\rangle_j$. So, since $y : \vec{B} \supset p \in \Gamma$, it suffices to show $(M, y\langle \mathcal{S}(\Gamma \Rightarrow B_i)\rangle_i) \in R_2$, which holds because $y : \vec{B} \supset p \in \Gamma$ and $\Gamma \vdash y\langle M'_i\rangle_i : p$ (the latter being a consequence of $y : \vec{B} \supset p \in \Gamma$, and $\Gamma \vdash M'_i : B_i$, for all i).

2. Take an arbitrary element of R_2. So, it must be of the form $(x\langle N_i\rangle_i, x\langle \mathcal{S}(\Gamma \Rightarrow B_i)\rangle_i)$ s.t. $(x : \vec{B} \supset p) \in \Gamma$ and $\Gamma \vdash x\langle N_i\rangle_i : p$. From the latter follows $\Gamma \vdash N_i : B_i$, for all i. So, by definition of R_1, $(N_i, \mathcal{S}(\Gamma \Rightarrow B_i)) \in R_1$, for all i.

"Only if". Follows from Lemmas 5 and 12. □

Example 14 *Let us consider the case of Peirce's law that is not valid intuitionistically. We have (for $p \neq q$):*

$$\mathcal{S}(\Rightarrow \mathsf{PEIRCE}) = \lambda x^{(p\supset q)\supset p}.x\langle \lambda y^p.\mathbb{O}\rangle$$

The fact that we arrived at $\mathbb{O}$ and found no elimination alternatives on the way annihilates *the co-term and implies there are no terms in the solution space of $\Rightarrow$ PEIRCE (hence no proofs, nor even infinite solutions).*

4 Co-contraction

In this section, consisting of three subsections, we introduce and study the *co-contraction* operation on Böhm forests. The main result of this section is Lemma 33, in the third subsection, because of its role in the proof of Theorem 54—the main theorem of the paper. Lemma 33 shows that co-contraction is the right operation to apply to a solution space $T = \mathcal{S}(\Gamma \Rightarrow C)$ to express the effect on the solution space of growing the context Γ to an *inessential extension* Γ'—this growth is made precise below and denoted by $\Gamma \leq \Gamma'$. Before, in the second subsection, the more general situation, where T is any Böhm forest (not necessarily a solution space) is analyzed in Lemma 24, a result that shows in what sense co-contraction witnesses the inversion of the inference rule of contraction. Finally, inversion of contraction is related to (and follows from) a kind of inversion of substitution, whose most general form is contained in Lemma 22, to be found already in the first subsection.

The co-contraction operation on Böhm forests, denoted $[\Gamma'/\Gamma]N$, is defined only when $\Gamma \leq \Gamma'$. Roughly speaking, the co-contraction effect at the level of Böhm forests is to add new elimination alternatives, made possible by the presence of more variables in Γ'. This effect is best seen in the last clause of Def. 16.

Definition 15 *1.* $|\Gamma| = \{A \mid \textit{there is } x \textit{ s. t. } (x : A) \in \Gamma\}$.

2. $\Gamma \leq \Gamma'$ *if* $\Gamma \subseteq \Gamma'$ *and* $|\Gamma| = |\Gamma'|$.

Notice that $|\Gamma|$ has only one element for each type occurring in the declarations of Γ. It thus abstracts away from multiple hypotheses of the same formula.

Definition 16 *Let* $\Gamma \leq \Gamma'$. *For* T *an expression of* $\overline{\lambda}^{co}_{\Sigma}$, *we define* $[\Gamma'/\Gamma]T$ *by corecursion as follows:*

$$\begin{array}{lcll} [\Gamma'/\Gamma](\lambda x^A.N) & = & \lambda x^A.[\Gamma'/\Gamma]N & \\ [\Gamma'/\Gamma]\sum_i E_i & = & \sum_i [\Gamma'/\Gamma]E_i & \\ [\Gamma'/\Gamma]\big(z\langle N_i\rangle_i\big) & = & z\langle[\Gamma'/\Gamma]N_i\rangle_i & \textit{if } z \notin \textit{dom}(\Gamma) \\ [\Gamma'/\Gamma]\big(z\langle N_i\rangle_i\big) & = & \sum_{(w:A)\in\Delta_z} w\langle[\Gamma'/\Gamma]N_i\rangle_i & \textit{if } z \in \textit{dom}(\Gamma) \end{array}$$

where $A := \Gamma(z)$ *and* $\Delta_z := \{(z : A)\} \cup (\Gamma' \setminus \Gamma)$.

The usual convention on bound variables applies, which requires in the first clause that the name x *is chosen so that it does not appear in* Γ'.

The effect of the last clause is to replace the summand $z\langle N_i\rangle_i$ with z of type $\Gamma(z)$ according to Γ with the sum of all $w\langle N_i\rangle_i$ that receive this type according to the potentially bigger context Γ', excluding the other variables of Γ but including the case $w = z$, and to continue the operation corecursively in the argument terms.[4]

Lemma 17 *If* $\mathsf{mem}(M, T)$ *and* $\Gamma \leq \Gamma'$ *then* $\mathsf{mem}(M, [\Gamma'/\Gamma]T)$.

Proof A coinductive proof can confirm the obvious intuition of the effect of co-contraction: either a summand is maintained, with corecursive application of co-contraction to the subterms, or it is replaced by a sum with even extra summands. □

Lemma 18 $[\Gamma/\Gamma]T = T$.

Proof Obvious coinduction for all expressions. □

We formally extend the co-contraction data from contexts to sequents σ. (This overloading of the operation will only be used in the next section.)

Definition 19 *Let* $\sigma = (\Gamma \Rightarrow A)$ *and* $\sigma' = (\Gamma' \Rightarrow A')$.

1. $\sigma \leq \sigma'$ *if* $\Gamma \leq \Gamma'$ *and* $A = A'$;

2. if $\sigma \leq \sigma'$, *then* $[\sigma'/\sigma]T := [\Gamma'/\Gamma]T$.

[4] In the workshop version [EMP13], we had a more "aggressive" version of co-contraction that did not exclude the other variables of Γ in the last clause, and for which we further added the binding $x : A$ to Γ and Γ' in the corecursive call in the λ-abstraction case. On solutions, these differences are immaterial, c. f. the example after Lemma 28.

4.1 Co-contraction and substitution

Co-contraction is a form of undoing substitution, in the following sense ($N \in \overline{\lambda}^{co}$):

$$\mathsf{mem}(N, [\Gamma, x : A, y : A/\Gamma, x : A][x/y]N) \tag{2}$$

In fact, we prove a stronger result. Let $[x/x_1, \cdots, x_n]N$ denote $[x/x_1] \cdots [x/x_n]N$.

Lemma 20 (Undo substitution) *For $N \in \overline{\lambda}^{co}$, $T \in \overline{\lambda}^{co}_{\Sigma}$,*

$$\mathsf{mem}([x_1/x_1, \cdots, x_n]N, T) \Rightarrow \mathsf{mem}(N, [\Gamma, x_1 : A, \ldots, x_n : A/\Gamma, x_1 : A]T) \ .$$

Proof Obviously, it suffices to show the statement with a term N' in place of the expression T. This will follow from R_1 below being included in the membership relation with terms as second argument. Let $\Delta := \Gamma, x_1 : A$ and $\Delta' := \Gamma, x_1 : A, \ldots, x_n : A$. Let

$$\begin{array}{lcl} R_1 & := & \{(N, [\Delta'/\Delta]N') \mid \mathsf{mem}([x_1/x_1, \cdots, x_n]N, N')\} \\ R_2 & := & \{(z\langle N_i\rangle_i, z\langle [\Delta'/\Delta]N_i'\rangle_i) \mid \forall i, \mathsf{mem}(N_i, [\Delta'/\Delta]N_i') \in R_1\} \end{array}$$

We argue by coinduction on membership. The proof obligations named (1)(a), (1)(b), and (2) in the proof of Proposition 13 are renamed here Ia, Ib, and II, respectively.

Let $(N, [\Delta'/\Delta]N') \in R_1$, hence

$$\mathsf{mem}([x_1/x_1, \cdots, x_n]N, N') \ . \tag{3}$$

We have to show that Ia or Ib holds. We proceed by case analysis of N.

Case $N = \lambda z.N_0$. Then $\mathsf{mem}(\lambda z.[x_1/x_1, \cdots, x_n]N_0, N')$, hence, by definition of membership, we must have $N' = \lambda z.N_0'$ and

$$\mathsf{mem}([x_1/x_1, \cdots, x_n]N_0, N_0') \ , \tag{4}$$

hence $[\Delta'/\Delta]N' = \lambda z.[\Delta'/\Delta]N_0'$. From (4) and definition of R_1 we get $(N_0, [\Delta'/\Delta]N_0') \in R_1$, so Ia holds.

Otherwise, that is, if N is not a λ-abstraction, then the same is true of $[x_1/x_1, \cdots, x_n]N$, hence (3) implies that $N' = \sum_j E_j'$, with

$$\mathsf{mem}([x_1/x_1, \cdots, x_n]N, E_j') \tag{5}$$

for some j, hence

$$[\Delta'/\Delta]N' = \sum_j [\Delta'/\Delta]E_j' \ . \tag{6}$$

To fulfil Ib, we need $(N, E) \in R_2$, for some summand E' of (6). From (5) and the definition of membership we must have $N = z\langle N_i\rangle_i$, for some z, hence

$$[x_1/x_1, \cdots, x_n]N = w\langle [x_1/x_1, \cdots, x_n]N_i\rangle_i \ , \tag{7}$$

with w a variable determined by z and $x_1, \cdots, x_n$ as follows: if $z \in \{x_1, \ldots, x_n\}$, then $w = x_1$, else $w = z$. Facts (5) and (7) give $E_j' = w\langle N_i'\rangle_i$ and, for all i,

$$\mathsf{mem}([x_1/x_1, \cdots, x_n]N_i, N_i') \ , \tag{8}$$

hence

$$(N_i, [\Delta'/\Delta]N_i') \in R_1 \ . \tag{9}$$

Now we will see that $z\langle [\Delta'/\Delta]N_i'\rangle_i$ is a summand of $[\Delta'/\Delta]E_j'$, sometimes the unique one. There are two cases:

First case: $z \in \{x_1, \ldots, x_n\}$. Then $[\Delta'/\Delta]E_j' = \sum_{k=1}^n x_k\langle [\Delta'/\Delta]N_i'\rangle_i$, since $w = x_1$.

Second case: otherwise, $w = z$. Now, by definition of co-contraction, $z\langle[\Delta'/\Delta]N'_i\rangle_i$ is always a summand of $[\Delta'/\Delta](z\langle N'_i\rangle_i)$, and the latter is $[\Delta'/\Delta]E'_j$ since $w = z$.

Therefore, $z\langle[\Delta'/\Delta]N'_i\rangle_i$ is a summand of sum (6). Moreover, $(N, z\langle[\Delta'/\Delta]N'_i\rangle_i) \in R_2$ by definition of R_2 and (9). So Ib holds.

Now let $(z\langle N_i\rangle_i, z\langle[\Delta'/\Delta]N'_i\rangle_i) \in R_2$. Proof obligation II is fulfilled, as $(N_i, [\Delta'/\Delta]N'_i) \in R_1$ holds for all i, by definition of R_2. □

Fact (2) follows from the previous lemma by taking $n = 2$, $x_1 = x$, $x_2 = y$ and $T = [x_1/x_1, x_2]N$.

The converse of the implication in Lemma 20 fails if other declarations with type A exist in Γ.

Example 21 *Let $\Gamma := \{z : A\}$, $\Delta := \Gamma, x : A$, $\Delta' := \Gamma, x : A, y : A$, $N := y$ and $T := z$. Then N is a member of $[\Delta'/\Delta]T$, since $[\Delta'/\Delta]T = z + y$, but $[x/y]N = x$ and x is not a member of T.*

The result of a co-contraction $[\Gamma, x_1 : A, \cdots, x_n : A/\Gamma, x_1 : A]T$, where Γ has no declarations with type A, does not depend on Γ nor A, so it deserves a lighter notation as $[x_1 + \cdots + x_n/x_1]T$. This particular case of the operation satisfies the equations:

$$\begin{array}{lcll}
[x_1 + \cdots + x_n/x_1](\lambda x^A.N) & = & \lambda x^A.[x_1 + \cdots + x_n/x_1]N & \\
[x_1 + \cdots + x_n/x_1]\sum_i E_i & = & \sum_i [x_1 + \cdots + x_n/x_1]E_i & \\
[x_1 + \cdots + x_n/x_1]\big(z\langle N_i\rangle_i\big) & = & z\langle[x_1 + \cdots + x_n/x_1]N_i\rangle_i & \text{if } z \neq x_1 \\
[x_1 + \cdots + x_n/x_1]\big(x_1\langle N_i\rangle_i\big) & = & \sum_{j=1}^n x_j\langle[x_1 + \cdots + x_n/x_1]N_i\rangle_i &
\end{array}$$

For this particular case, we get a pleasing formula:

Lemma 22 (Undo substitution) *For $N \in \overline{\lambda}^{co}$, $T \in \overline{\lambda}^{co}_{\Sigma}$,*

$$\mathsf{mem}([x_1/x_1, \cdots, x_n]N, T) \Leftrightarrow \mathsf{mem}(N, [x_1 + \cdots + x_n/x_1]T) \ ,$$

provided $x_i \notin FV(T)$, $i = 2, \ldots, n$.

Proof "Only if". Particular case of Lemma 20.

"If". Let $\phi(T)$ denote the proviso on T. Let

$$\begin{array}{lcl}
R_1 & := & \{([x_1/x_1, \cdots, x_n]N, N') \mid \phi(N') \wedge \mathsf{mem}(N, [x_1 + \cdots + x_n/x_1]N')\} \\
R_2 & := & \{(z\langle[x_1/x_1, \cdots, x_n]N_i\rangle_i, z\langle N'_i\rangle_i) \mid \forall i, ([x_1/x_1, \cdots, x_n]N_i, N'_i) \in R_1\}
\end{array}$$

We argue by coinduction on membership and thus obtain the "if" part with T replaced by N', from which the general case immediately follows. The proof obligations named (1)(a), (1)(b), and (2) in the proof of Proposition 13 are renamed here Ia, Ib, and II, respectively.

Let $([x_1/x_1, \cdots, x_n]N, N') \in R_1$, hence $\phi(N')$ and

$$\mathsf{mem}(N, [x_1 + \cdots + x_n/x_1]N') \ . \tag{10}$$

The proof proceeds by case analysis of N.

Case $N = \lambda z.N_0$, so $[x_1/x_1, \cdots, x_n]N = \lambda z.[x_1/x_1, \cdots, x_n]N_0$. By (10) and definitions of membership and of $[x_1 + \cdots + x_n/x_1]N'$, $N' = \lambda z.N'_0$, hence $\phi(N'_0)$ (because z is not one of $x_2, \cdots, x_n$), $[x_1 + \cdots + x_n/x_1]N' = \lambda z.[x_1 + \cdots + x_n/x_1]N'_0$ and

$$\mathsf{mem}(N_0, [x_1 + \cdots + x_n/x_1]N'_0) \ . \tag{11}$$

So $([x_1/x_1, \cdots, x_n]N_0, N'_0) \in R_1$, by definition of R_1, (11) and $\phi(N'_0)$, which completes proof obligation Ia.

Case $N = z\langle N_i\rangle_i$. Then $[x_1/x_1, \cdots, x_n]N = y\langle[x_1/x_1, \cdots, x_n]N_i\rangle_i$, with $y = x_1$ when $z \in \{x_1, \ldots, x_n\}$, and $y = z$ otherwise. From (10) and definitions of membership and of $[x_1 + \cdots + x_n/x_1]N'$, one gets $N' = \sum_j E'_j$, hence $\phi(E'_j)$ for all j, and $[x_1 + \cdots + x_n/x_1]N' = \sum_j [x_1 + \cdots + x_n/x_1]E'_j$.

In order to fulfil proof obligation Ib, we need $([x_1/x_1, \cdots, x_n]N, E') \in R_2$, for some summand E' of N'. From (10) again, we get, for some j,

$$\mathsf{mem}(z\langle N_i\rangle_i, [x_1 + \cdots + x_n/x_1]E'_j) \ . \tag{12}$$

Let $E'_j = w\langle N'_i\rangle_i$, hence $\phi(N'_i)$ for all i. We now have two cases:

First case: $w = x_1$. Then $[x_1 + \cdots + x_n/x_1]E'_j = \sum_{k=1}^n x_k\langle [x_1 + \cdots + x_n/x_1]N'_i\rangle_i$. From (12) we get, for some k,

$$\mathsf{mem}(z\langle N_i\rangle_i, x_k\langle [x_1 + \cdots + x_n/x_1]N'_i\rangle_i) \tag{13}$$

hence, for all i,

$$\mathsf{mem}(N_i, [x_1 + \cdots + x_n/x_1]N'_i) \ . \tag{14}$$

From (13), $z = x_k$, hence $y = x_1$. We prove $([x_1/x_1, \cdots, x_n]N, E'_j) \in R_2$, that is

$$(x_1\langle [x_1/x_1, \cdots, x_n]N_i\rangle_i, x_1\langle N'_i\rangle_i) \in R_2 \ .$$

By definition of R_2, we need $([x_1/x_1, \cdots, x_n]N_i, N'_i) \in R_1$, for all i. This follows from (14), $\phi(N'_i)$ and the definition of R_1.

Second case: $w \neq x_1$. Then $[x_1 + \cdots + x_n/x_1]E'_j = w\langle [x_1 + \cdots + x_n/x_1]N'_i\rangle_i$. From (12), $z = w$; from $\phi(E'_j)$ and $w \neq x_1$, $z \notin \{x_1, \ldots, x_n\}$. Still from (12), we get again (14) and now $([x_1/x_1, \cdots, x_n]N, E'_j) = (z\langle [x_1/x_1, \cdots, x_n]N_i\rangle_i, z\langle N'_i\rangle_i) \in R_2$ follows as before.

Let $(z\langle [x_1/x_1, \cdots, x_n]N_i\rangle_i, z\langle N'_i\rangle_i) \in R_2$, hence proof obligation II holds by definition of R_2. □

The proviso about variables $x_2, \cdots, x_n$ in the previous lemma is necessary for the "if" implication. Otherwise, one has the following counter-example: $n := 2$, $N := x_2$, and $T = x_2$. N is a member of $[x_1 + x_2/x_1]T = x_2$ but $x_1 = [x_1/x_1, x_2]N$ is not a member of T.

4.2 Co-contraction and contraction

Co-contraction is related to the inference rule of contraction. By *contraction* we mean the rule in the following lemma.

Lemma 23 (Contraction) *In $\overline{\lambda}$ the following rule is admissible and invertible:*

$$\frac{\Gamma, x : A, y : A \vdash t : B}{\Gamma, x : A \vdash [x/y]t : B} \ .$$

That is: for all $t \in \overline{\lambda}$, $\Gamma, x : A, y : A \vdash t : B$ iff $\Gamma, x : A \vdash [x/y]t : B$.

Proof Routine induction on t, using inversion of *RIntro* and *LVecIntro*. □

If $\Gamma \leq \Gamma'$, then, from a proof of $\Gamma' \Rightarrow B$, we get a proof of $\Gamma \Rightarrow B$ by a number of contractions. The following result justifies the terminology "co-contraction".

Lemma 24 (Co-contraction and types) *Let T be an expression of $\overline{\lambda}^{co}_{\Sigma}$ and $\Gamma' \cup \Delta$ be a context. If $\Gamma \cup \Delta \vdash T : B$ and $\Gamma \leq \Gamma'$ then $\Gamma' \cup \Delta \vdash [\Gamma'/\Gamma]T : B$.*

Proof (Notice that we exceptionally consider not necessarily disjoint unions of contexts. This is immaterial for the proof but will be needed in Lemma 48.) Immediate by coinduction.[5] □

In particular, if $\Gamma \vdash u : B$ in $\overline{\lambda}$ and $\Gamma \leq \Gamma'$, then indeed $\Gamma' \vdash [\Gamma'/\Gamma]u : B$ — but $[\Gamma'/\Gamma]u$ is not guaranteed to be a proof (*i. e.*, a term in $\overline{\lambda}$).

[5] With this lemma in place, invertibility in Lemma 23 follows from general reasons. Take $N = t$ in fact (2) and then apply this lemma and Lemma 12.

Example 25 *Let* $\Gamma := \{f : p \supset p \supset q, x : p\}$, $\Gamma' := \{f : p \supset p \supset q, x : p, y : p\}$, *and* $u := f\langle x, x\rangle$, *hence* $\Gamma \leq \Gamma'$ *and* $\Gamma \vdash u : q$. *Then,* $[\Gamma'/\Gamma]u = f\langle x + y, x + y\rangle$, *and the given particular case of the previous lemma entails* $\Gamma' \vdash f\langle x + y, x + y\rangle : q$. *The term* $f\langle x + y, x + y\rangle$ *is no* $\overline{\lambda}$*-term, but rather has several members. Due to Lemma 22, these are exactly the (four, in this case)* $t \in \overline{\lambda}$ *such that* $[x/y]t = u$. *Thanks to Lemma 23, it follows that each member* t *of* $f\langle x + y, x + y\rangle$ *satisfies* $\Gamma' \vdash t : q$.

On the other hand, if T in Lemma 24 is the solution space $\mathcal{S}(\Gamma \Rightarrow B)$ (rather than a mere member u of it), then $[\Gamma'/\Gamma]T$ is indeed the solution space $\mathcal{S}(\Gamma' \Rightarrow B)$ — but we have to wait until Lemma 33 to see the proof.

Example 26 *Continuing Example 25, since* $\mathcal{S}(\Gamma \Rightarrow q) = u$, *one has* $[\Gamma'/\Gamma]\mathcal{S}(\Gamma \Rightarrow q) = f\langle x + y, x + y\rangle$. *Lemma 33 will guarantee that* $f\langle x+y, x+y\rangle$ *(a term obtained from* u *by co-contraction) is the solution space* $\mathcal{S}(\Gamma' \Rightarrow q)$. *Thanks to Proposition 13, one sees again that each member of* t *of* $f\langle x + y, x + y\rangle$ *satisfies* $\Gamma' \vdash t : q$.

4.3 Co-contraction and solution spaces

The intuitive idea of the next notion is to capture saturation of sums, so to speak.

Definition 27 (Maximal co-contraction) *Let* $T \in \overline{\lambda}^{co}_{\Sigma}$ *and* Γ *be a context.*

1. *Consider an occurrence of* x *in* T. *Consider the traversed* λ*-abstractions from the root of* T *to the given occurrence of* x, *and let* $y_1^{A_1}, \ldots, y_n^{A_n}$ *be the respective variables. We call* $\Gamma, y_1 : A_1 \ldots, y_n : A_n$ *the* local extension *of* Γ *for the given occurrence of* x.
2. T *in* $\overline{\lambda}^{co}_{\Sigma}$ *is maximally co-contracted w. r. t.* Γ *if:*
   (a) *all free variables of* T *are declared in* Γ*; and*
   (b) *every occurrence of a variable* x *in* T *is as head of a summand* $x\langle N_i\rangle_i$ *in a sum in which also* $y\langle N_i\rangle_i$ *is a summand (modulo bisimilarity), for every variable* y *that gets the same type as* x *in the local extension of* Γ *for the occurrence of* x.

Lemma 28 (Solution spaces are maximally co-contracted) *Given sequent* $\Gamma \Rightarrow C$, *the solution space* $\mathcal{S}(\Gamma \Rightarrow C)$ *is maximally co-contracted w. r. t.* Γ.

Proof By coinduction. For the variable occurrences that are on display in the one-line formula (1) for $\mathcal{S}(\Gamma \Rightarrow \vec{A} \supset p)$—that is, for each of the y's that are head variables of the displayed summands—the local context is $\Delta = \Gamma, \vec{x} : \vec{A}$, and if y_1 and y_2 have the same type in Δ with target atom p, both variables appear as head variables with the same lists of argument terms. For variable occurrences hidden in the j-th argument of some y, we use two facts: (i) the j-th argument is maximally co-contracted w. r. t. Δ by coinductive hypothesis; (ii) Δ collects the variables λ-abstracted on the path from the root of the term to the root of j-th argument. □

Example 29 *Let* $\Gamma := \{z : p\}$, $\Delta := \Gamma, x : p$, $N := \lambda x^p.z\langle\rangle$ *and* $N' := \lambda x^p.z\langle\rangle + x\langle\rangle$. *The term* N *is not maximally co-contracted w. r. t.* Γ. *Intuitively, the sum* $z\langle\rangle$ *is not saturated, as it does not record all the alternative proofs of* $\Delta \Rightarrow p$. *Hence* N *cannot be the solution space* $\mathcal{S}(\Gamma \Rightarrow p \supset p)$ *— the latter is* N', *hence* N' *is maximally co-contracted w. r. t.* Γ, *by the previous lemma. The output of co-contraction* $[\Gamma/\Gamma]N$ *(being* N*) is not maximally co-contracted*[6]. *We will be interested mostly in applying co-contraction to already maximally co-contracted terms, e.g. solution spaces.*

Lemma 30 *If* $|\Gamma' \setminus \Gamma|$ *and* $|\Delta|$ *are disjoint,* Γ', Δ *is a context and* $\Gamma \leq \Gamma'$ *then* $[\Gamma', \Delta/\Gamma, \Delta]T = [\Gamma'/\Gamma]T$.

[6]This is in contrast with the definition of co-contraction in [EMP13], which outputs maximally co-contracted terms, e.g. $[\Gamma/\Gamma]N = N'$ in this case.

Proof Easy coinduction. □

The disjointness condition of the previous lemma is rather severe. It can be replaced by maximal co-contraction of the given term.

Lemma 31 *If Γ', Δ is a context, $\Gamma \leq \Gamma'$ and T is maximally co-contracted w. r. t. Γ, Δ, then $[\Gamma', \Delta/\Gamma, \Delta]T = [\Gamma'/\Gamma]T$.*

Proof By coinduction. The proof then boils down to showing for any subterm $z\langle N_i\rangle_i$ of T, if a $w \neq z$ is found according to the last clause of the definition of co-contraction with $[\Gamma', \Delta/\Gamma, \Delta]$, then one can also find w according to the last clause of the definition of co-contraction with $[\Gamma'/\Gamma]$. Assume such a w. Since it comes from the last clause, we have $z \in dom(\Gamma, \Delta)$ (hence, by the usual convention on the naming of bound variables, z is even a free occurrence in T), and $(w : (\Gamma, \Delta)(z)) \in \Gamma' \setminus \Gamma$. If $z \in dom(\Gamma)$, then we are obviously done. Otherwise, $z \in dom(\Delta)$, and so $(w : \Delta(z)) \in \Gamma' \setminus \Gamma$. Since $|\Gamma'| = |\Gamma|$, there is $(x : \Delta(z)) \in \Gamma$. Since T is maximally co-contracted w. r. t. Γ, Δ, the subterm $z\langle N_i\rangle_i$ is one summand in a sum which also has the summand $x\langle N_i\rangle_i$, and for the latter summand, the last clause of the definition of co-contraction with $[\Gamma'/\Gamma]$ can be used with $(w : \Gamma(x)) \in \Gamma' \setminus \Gamma$. □

Corollary 32 *If Γ', Δ is a context, $\Gamma \leq \Gamma'$, then $[\Gamma', \Delta/\Gamma, \Delta]\mathcal{S}(\Gamma, \Delta \Rightarrow C) = [\Gamma'/\Gamma]\mathcal{S}(\Gamma, \Delta \Rightarrow C)$.*

Proof Combine the preceding lemma with Lemma 28.[7] □

The following main result of this section says that the solution space w. r. t. an inessential extension of a context is obtained by applying the co-contraction operation to the solution space corresponding to the original context.

Lemma 33 (Co-contraction and solution spaces) *If $\Gamma \leq \Gamma'$ then we have $\mathcal{S}(\Gamma' \Rightarrow C) = [\Gamma'/\Gamma](\mathcal{S}(\Gamma \Rightarrow C))$.*

Proof Let $R := \{(\mathcal{S}(\Gamma' \Rightarrow C), [\Gamma'/\Gamma](\mathcal{S}(\Gamma \Rightarrow C))) \mid \Gamma \leq \Gamma', C \text{ arbitrary}\}$. We prove that R is closed backward relative to the notion of bisimilarity taking sums of alternatives as if they were sets. From this, we conclude $R \subseteq =$.

$$\mathcal{S}(\Gamma' \Rightarrow C) = \lambda z_1^{A_1} \cdots z_n^{A_n}. \sum_{(z:\vec{B}\supset p)\in\Delta'} z\langle \mathcal{S}(\Delta' \Rightarrow B_j)\rangle_j \tag{15}$$

and

$$[\Gamma'/\Gamma](\mathcal{S}(\Gamma \Rightarrow C)) = \lambda z_1^{A_1} \cdots z_n^{A_n}. \sum_{(y:\vec{B}\supset p)\in\Delta} \sum_{(w:\Delta(y))\in\Delta'_y} w\langle [\Gamma'/\Gamma]\mathcal{S}(\Delta \Rightarrow B_j)\rangle_j \tag{16}$$

where $\Delta := \Gamma, z_1 : A_1, \ldots, z_n : A_n$, $\Delta' := \Gamma', z_1 : A_1, \ldots, z_n : A_n$, for $y \in dom(\Gamma)$, $\Delta'_y := \{(y : \Delta(y))\} \cup (\Gamma' \setminus \Gamma)$, and for $y = z_i$, $\Delta'_y = \{(y : \Delta(y))\}$.

From $\Gamma \leq \Gamma'$ we get $\Delta \leq \Delta'$, hence

$$(\mathcal{S}(\Delta' \Rightarrow B_j), [\Delta'/\Delta]\mathcal{S}(\Delta \Rightarrow B_j)) \in R \ ,$$

which fits with the summands in (16) since, by Corollary 32, $[\Delta'/\Delta]\mathcal{S}(\Delta \Rightarrow B_j) = [\Gamma'/\Gamma]\mathcal{S}(\Delta \Rightarrow B_j)$. To conclude the proof, it suffices to show that (i) each head-variable z that is a "capability" of the summation in (15) is matched by a head-variable w that is a "capability" of the summation in (16); and (ii) vice-versa.

(i) Let $z \in dom(\Delta')$. We have to exhibit $y \in dom(\Delta)$ such that $(z : \Delta(y)) \in \Delta'_y$. First case: $z \in dom(\Delta)$. Then, $(z : \Delta(z)) \in \Delta'_z$. So we may take $y = z$. Second and last case:

[7] The notion of being maximally co-contracted is not essential for this paper. Only this corollary will be used in the sequel, and it could also be proven directly, in the style of the proof of the following lemma. For this to work smoothly, the statement should be generalized to: If Γ', Δ, Θ is a context, $\Gamma \leq \Gamma'$, then $[\Gamma', \Delta/\Gamma, \Delta]\mathcal{S}(\Gamma, \Delta, \Theta \Rightarrow C) = [\Gamma'/\Gamma]\mathcal{S}(\Gamma, \Delta, \Theta \Rightarrow C)$.

$z \in dom(\Gamma') \setminus dom(\Gamma)$. By definition of $\Gamma \leq \Gamma'$, there is $y \in dom(\Gamma)$ such that $(z : \Gamma(y)) \in \Gamma'$. Since $\Gamma(y) = \Delta(y)$ and $z \notin dom(\Delta)$, we get $(z : \Delta(y)) \in \Delta'_y$.

(ii) We have to show that, for all $y \in dom(\Delta)$, and all $(w : \Delta(y)) \in \Delta'_y$, $(w : \Delta(y)) \in \Delta'$. But this is immediate. □

Notice that we cannot expect that the summands appear in the same order in (15) and (16). Therefore, we are obliged to use symmetry of $+$. It is even convenient to disregard multiplicity, as seen in the following example.

Example 34 *Let $\Gamma := x : p$, $\Gamma' := \Gamma, y : p$, $\Delta := z : p$, $\Theta := \Gamma, \Delta$, $\Theta' := \Gamma', \Delta$ and $C := p$. Then $\mathcal{S}(\Theta \Rightarrow C) = x + z$ and $\mathcal{S}(\Theta' \Rightarrow C) = x + y + z$. This yields $[\Theta'/\Theta]\mathcal{S}(\Theta \Rightarrow C) = (x + y) + (z + y)$ and $[\Gamma'/\Gamma]\mathcal{S}(\Theta \Rightarrow C) = (x + y) + z$, where parentheses are only put to indicate how co-contraction has been calculated. Taken together, these calculations contradict the strengthening of Lemma 33 without idempotence of $+$, when the parameters Γ, Γ', of the lemma are taken as Θ, Θ', and they also contradict the analogous strenghtening of Corollary 32 when the parameters Γ, Γ', Δ, C of the corollary are as given here.*

The summand-wise and therefore rather elegant definition of co-contraction is the root cause for this blow-up of the co-contracted terms. However, mathematically, there is no blow-up since we identify $(x + y) + (z + y)$ with $x + y + z$, as they represent the same set of elimination alternatives.

In the light of Lemma 28, Lemma 33 shows that $\mathcal{S}(\Gamma \Rightarrow C)$, which is maximally co-contracted w. r. t. Γ, only needs the application of the co-contraction operation $[\Gamma'/\Gamma]$ for $\Gamma \leq \Gamma'$ to obtain a term that is maximally co-contracted w. r. t. Γ'.

Example 35 (Example 10 continued) *Thanks to Lemma 33, N_9 is obtained by co-contraction from N_5:*

$$N_9 = [x : \cdot, y : (p \supset q) \supset p, z : p, y_1 : (p \supset q) \supset p, z_1 : p \,/\, x : \cdot, y : (p \supset q) \supset p, z : p]N_5 \ ,$$

where the type of x has been omitted. Hence, N_6, N_7, N_8 and N_9 can be eliminated, and N_5 can be expressed as the (meta-level) fixed point:

$$N_5 = \nu\, N.x\langle\lambda y_1^{(p\supset q)\supset p}.y\langle\lambda z_1^p.[x, y, z, y_1, z_1/x, y, z]N\rangle + z + y_1\langle\lambda z_1^p.[x, y, z, y_1, z_1/x, y, z]N\rangle\rangle \ ,$$

now missing out all types in the co-contraction operation(s). Finally, we obtain the closed Böhm forest

$$\mathcal{S}(\Rightarrow \mathsf{DNPEIRCE}) = \lambda x^{\mathsf{PEIRCE}\supset q}.x\langle\lambda y^{(p\supset q)\supset p}.y\langle\lambda z^p.N_5\rangle\rangle$$

This representation also makes evident that, by exploiting the different co-contracted copies of y, there are infinitely many $M \in \overline{\lambda}^{co} \setminus \overline{\lambda}$ such that $\mathsf{mem}(M, \mathcal{S}(\Rightarrow \mathsf{DNPEIRCE}))$, in other words, $\Rightarrow \mathsf{DNPEIRCE}$ has infinitely many infinite solutions.

Example 36 (Example 11 continued) *Likewise, Lemma 33 shows that, with the notation of Ex. 11 and omitting the types in the co-contraction operation, $N' = [x, y, z/x, y]N$, hence*

$$\mathcal{S}(\Rightarrow \mathsf{THREE}) = \lambda x^{(p\supset p)\supset p}.x\langle\lambda y^p.\nu N.x\langle\lambda z^p.[x, y, z/x, y]N\rangle + y\rangle$$

Visibly, the only infinite solution is obtained by choosing always the left alternative, creating infinitely many vacuous bindings, thus it can be described as $\lambda x^{(p\supset p)\supset p}.N_0$ with $N_0 = x\langle\lambda\_^p.N_0\rangle$ (where $\_$ is the name of choice for a variable that has no bound occurrences).

We have now seen succinct presentations of the solution spaces of all of the examples in Ex. 1. Although described with few mathematical symbols, they are still on the informal level of infinitary terms with meta-level fixed points, but this will be remedied by a finitary system in the next section.

5 A typed finitary system for solution spaces

Here, we develop a finitary lambda-calculus to represent solution spaces of proof search problems in $\overline{\lambda}$. The main points in the design of the calculus are:

1. Fixed-point variables stand for (spaces of) solutions;
2. Fixed-point variables are typed by sequents;
3. A relaxed form of binding of fixed-point variables has to be allowed, and controlled through the typing system.

There is a sound semantics of the typed finitary terms into Böhm forests, which is complete w.r.t. those Böhm forests that represent solution spaces. The relaxed form of binding is matched, on the semantical side, by the special operation of co-contraction.

5.1 The untyped system $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$

The set of inductive cut-free $\overline{\lambda}$-terms with finite numbers of elimination alternatives, and a fixed-point operator is denoted by $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$ and is given by the following grammar (read inductively):

$$\begin{array}{lrcl} \text{(terms)} & N & ::= & \lambda x^A.N \mid \mathsf{gfp}\ X^\sigma.E_1 + \cdots + E_n \mid X^\sigma \\ \text{(elim. alternatives)} & E & ::= & x\langle N_1, \ldots, N_k\rangle \end{array}$$

where X is assumed to range over a countably infinite set of *fixed-point variables* (also letters Y, Z will range over them) that may also be thought of as meta-variables, and where, as for $\overline{\lambda}^{co}_{\Sigma}$, both $n, k \geq 0$ are arbitrary. We extend our practice established for $\overline{\lambda}^{co}_{\Sigma}$ of writing the sums $E_1 + \cdots + E_n$ in the form $\sum_i E_i$ for $n \geq 0$. Also the tuples continue to be communicated as $\langle N_i\rangle_i$. As for $\overline{\lambda}^{co}_{\Sigma}$, we will identify expressions modulo symmetry and idempotence of $+$, thus treating sums of elimination alternatives as if they were the set of those elimination alternatives. Again, we will write T for expressions of $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$, i. e., for terms and elimination alternatives.

In the term formation rules, sequents σ appear. We require them to be *atomic*, i. e., of the form $\Gamma \Rightarrow p$ with atomic conclusion. Let $FPV(T)$ denote the set of free occurrences of typed fixed-point variables in T. Perhaps unexpectedly, in $\mathsf{gfp}\ X^\sigma.\sum_i E_i$ the fixed-point construction gfp binds *all* free occurrences of $X^{\sigma'}$ in the elimination alternatives E_i, not just X^σ. But we only want this to happen when $\sigma \leq \sigma'$. In fact, the sequent σ serves a different purpose than being the type of fixed-point variable X, see below on well-bound expressions.

In the sequel, when we refer to *finitary terms* we have in mind the terms of $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$. The fixed-point operator is called gfp ("greatest fixed point") to indicate that its semantics is (now) defined in terms of infinitary syntax, but there, fixed points are unique. Hence, the reader may just read this as "the fixed point".

We next present the interpretation of expressions of $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$ in terms of the coinductive syntax of $\overline{\lambda}^{co}_{\Sigma}$ (using the ν operation on the meta-level), which is more precise on the conditions that guarantee its well-definedness than the interpretation of finitary terms introduced in [EMP13]. (Nonetheless, in the cited paper, no problem arises with the less precise definitions since only representations of solution spaces were interpreted, see below.)

We call an expression T *trivially regular* if $FPV(T)$ has no duplicates: A set S of typed fixed-point variables is said to *have no duplicates* if the following holds: if $X^{\sigma_1}, X^{\sigma_2} \in S$, then $\sigma_1 = \sigma_2$. We *do not* confine our investigation to trivially regular expressions, see Appendix A for an example where we require more flexibility.

Definition 37 (regularity in $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$) *Let $T \in \overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$. T is regular if for all fixed-point variable names X, the following holds: if $X^\sigma \in FPV(T)$ for some sequent σ, then there is a sequent σ_0 such that, for all $X^{\sigma'} \in FPV(T)$, $\sigma_0 \leq \sigma'$.*

Obviously, every trivially regular T is regular (using $\sigma_0 := \sigma$ and reflexivity of $\leq$ since $\sigma' = \sigma$). Trivially, every closed T, i. e., with $FPV(T) = \emptyset$, is trivially regular.

Interpretation of expressions of $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$ is done with the help of *environments*, a notion which will be made more precise than in [EMP13]. Since interpretations of T only depend on the values of the environment on $FPV(T)$, we rather assume that environments are partial functions with a finite domain. Hence, an environment ξ is henceforth a partial function from typed fixed-point variables X^σ to (co)terms of $\overline{\lambda}_\Sigma^{co}$ with finite domain $dom(\xi)$ that has no duplicates (in the sense made precise above).

The interpretation function will also be made partial: $[\![T]\!]_\xi$ will only be defined when environment ξ is admissible for T:

Definition 38 (admissible environment) *An environment ξ is admissible for expression T of $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$ if for every $X^{\sigma'} \in FPV(T)$, there is an $X^\sigma \in dom(\xi)$ such that $\sigma \leq \sigma'$.*

Notice that the required sequent σ in the above definition is unique since ξ is supposed to be an environment. This observation even implies the following characterization of regularity:

Lemma 39 *$T \in \overline{\lambda}_\Sigma^{\mathsf{gfp}}$ is regular iff there is an environment ξ that is admissible for T.*

Proof Obvious. □

We have to add a further restriction before defining the interpretation function:

Definition 40 (well-bound expression) *We call an expression T of $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$* well-bound *iff for any of its subterms $\mathsf{gfp}\ X^\sigma.\sum_i E_i$ and any (free) occurrence of $X^{\sigma'}$ in the E_i's, $\sigma \leq \sigma'$.*

Definition 41 (interpretation of finitary terms as Böhm forests) *For a well-bound expression T of $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$, the interpretation $[\![T]\!]_\xi$ for an environment ξ that is admissible for T is given by structural recursion on T:*

$$\begin{array}{rcl} [\![X^{\sigma'}]\!]_\xi & = & [\sigma'/\sigma]\xi(X^\sigma) \quad \textit{for the unique } \sigma \leq \sigma' \textit{ with } X^\sigma \in dom(\xi) \\ [\![\mathsf{gfp}\ X^\sigma.\sum_i E_i]\!]_\xi & = & \nu\, N.\sum_i [\![E_i]\!]_{\xi \cup [X^\sigma \mapsto N]} \\ [\![\lambda x^A.N]\!]_\xi & = & \lambda x^A.[\![N]\!]_\xi \\ [\![x\langle N_i\rangle_i]\!]_\xi & = & x\langle [\![N_i]\!]_\xi\rangle_i \end{array}$$

Notice that the case of gfp uses the extended environment $\xi \cup [X^\sigma \mapsto N]$ that is admissible for E_i thanks to our assumption of well-boundness. (Moreover, by renaming X, we may suppose that there is no $X^{\sigma'}$ in $dom(\xi)$.). The meta-level fixed point over N is well-formed since every elimination alternative starts with a head/application variable, and all occurrences of N in the summands are thus *guarded* by constructors for elimination alternatives, and therefore the fixed-point definition is *productive* (in the sense of producing more and more data of the fixed point through iterated unfolding) and uniquely determines a Böhm forest, unlike an expression of the form $\nu\, N.N$ that does not designate a Böhm forest and would only come from the syntactically illegal term $\mathsf{gfp} X^\sigma.X^\sigma$.

The interpretation $[\![T]\!]_\xi$ only depends on the values of ξ for arguments X^σ for which there is a sequent σ' such that $X^{\sigma'} \in FPV(T)$. In more precise words, the interpretations $[\![T]\!]_\xi$ and $[\![T]\!]_{\xi'}$ coincide whenever ξ and ξ' have the same domain and agree on all typed fixed-point variables X^σ for which there is a sequent σ' such that $X^{\sigma'} \in FPV(T)$.

If T is closed, i. e., $FPV(T) = \emptyset$, then the empty function is an admissible environment for T, and the environment index in the interpretation is left out, hence the interpretation is abbreviated to $[\![T]\!]$. Anyway, the interpretation of a closed T does not depend on the environment.

If no $X^{\sigma'}$ occurs free in $\sum_i E_i$ for any sequent σ', we allow ourselves to abbreviate the finitary term $\mathsf{gfp}\ X^\sigma.\sum_i E_i$ as $\sum_i E_i$. Thanks to our observation above on the dependence of $[\![T]\!]_\xi$ on ξ, we have $[\![\sum_i E_i]\!]_\xi = \sum_i [\![E_i]\!]_\xi$.

Figure 5: Typing system for $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$

$$\frac{(X:\sigma)\in\Xi \qquad \sigma\le\sigma'=(\Theta'\Rightarrow p) \qquad \Theta'\subseteq\Gamma}{\Xi\rfloor\Gamma\vdash X^{\sigma'}:p}$$

$$\frac{\text{for all } i,\ \Xi, X:\sigma\rfloor\Gamma\vdash E_i:p \qquad \sigma=(\Theta\Rightarrow p) \qquad \Theta\subseteq\Gamma}{\Xi\rfloor\Gamma\vdash \mathsf{gfp}\, X^{\sigma}.\textstyle\sum_i E_i:p}$$

$$\frac{\Xi\rfloor\Gamma, x:A\vdash N:B}{\Xi\rfloor\Gamma\vdash \lambda x^A.N:A\supset B} \qquad \frac{(x:\vec{B}\supset p)\in\Gamma \qquad \text{for all } i,\ \Xi\rfloor\Gamma\vdash N_i:B_i}{\Xi\rfloor\Gamma\vdash x\langle N_i\rangle_i:p}$$

5.2 Typing system for $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$

The typing system for $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$ is defined in Figure 5. The main desiderata for this typing system were to be able to prove Lemma 48 that the interpretation of finitary terms as Böhm forests preserves types and Lemma 53 that a finitary representation of the solution space (that can be found) has the original sequent as type—for details see below.

The typing system for $\overline{\lambda}_\Sigma^{\mathsf{gfp}}$ derives sequents $\Xi\rfloor\Gamma\vdash T:B$. The first context Ξ has the form $\overrightarrow{X:\sigma}$, so fixed-point variables are typed by sequents. The first typing rule in Figure 5 implies that fixed-point variables enjoy a relaxed form of binding.

The context Ξ is such that no fixed-point variable name X occurs twice (there is no condition concerning duplication of sequents). So, Ξ can be (and will be) seen as a partial function, and Ξ, when regarded as a set of typed fixed-point variables, has no duplicates. If Ξ is empty, then we write $\Gamma\vdash T:B$ instead of $\Xi\rfloor\Gamma\vdash T:B$.

Lemma 42 *If $\Xi\rfloor\Gamma\vdash T:B$, $\Xi\subseteq\Xi'$ and $\Gamma\subseteq\Gamma'$ then $\Xi'\rfloor\Gamma'\vdash T:B$.*

Proof Obvious since, for the Ξ argument, there is only look-up, and for the Γ argument, weakening is directly built into the rules concerning fixed-point variables and goes through inductively for the others. □

Lemma 43 *If $\Xi\rfloor\Gamma\vdash T:B$ then the free variables of T are in $dom(\Gamma)$.*

Notice that the free variables of $X^{\Gamma\Rightarrow p}$ are $dom(\Gamma)$ and that $dom(\Gamma)$ enters the free variables of $\mathsf{gfp}\, X^{\Gamma\Rightarrow p}.\sum_i E_i$.

Proof Induction on T. □

Lemma 44 *If $\Xi\rfloor\Gamma\vdash T:B$ and $X^{\sigma'}\in FPV(T)$ then there is a sequent σ such that $(X:\sigma)\in\Xi$ and $\sigma\le\sigma'$.*

Proof Induction on T. □

Corollary 45 *If $\Xi\rfloor\Gamma\vdash T:B$, and ξ is a partial function from typed fixed-point variables X^σ to (co)terms of $\overline{\lambda}_\Sigma^{co}$ with domain Ξ, then ξ is an environment, and it is admissible for T.*

As a consequence of the last lemma, we obtain by induction on T:

Lemma 46 (Typable terms are well-bound) *If $\Xi\rfloor\Gamma\vdash T:B$ then T is well-bound.*

Proof Induction on T. □

Definition 47 (Well-typed environment) *An environment ξ is well-typed w. r. t. context Γ if for all $X^{\Theta\Rightarrow q}\in dom(\xi)$, $\Theta\subseteq\Gamma$ and $\Gamma\vdash\xi(X^{\Theta\Rightarrow q}):q$ (in $\overline{\lambda}_\Sigma^{co}$).*

Lemma 48 (Interpretation preserves types) *Let $\Xi \rfloor \Gamma \vdash T : B$ in $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$ and ξ be a well-typed environment w. r. t. Γ with $dom(\xi) = \Xi$. Then $\Gamma \vdash [\![T]\!]_\xi : B$ in $\overline{\lambda}^{co}_{\Sigma}$. In particular, if $\Gamma \vdash T : B$ in $\overline{\lambda}^{\mathsf{gfp}}_{\Sigma}$, then $\Gamma \vdash [\![T]\!] : B$ in $\overline{\lambda}^{co}_{\Sigma}$.*

Proof Induction on T, using Lemma 24 in the base case of a fixed-point variable and using an embedded coinduction in the case of a greatest fixed point. □

5.3 Finitary representation of solution spaces

Solution spaces for $\overline{\lambda}$ can be shown to be finitary, with the help of the *finitary representation mapping* $\mathcal{F}(\sigma; \Xi)$, which we introduce now.

Definition 49 *Let $\Xi := \overrightarrow{X : \Theta \Rightarrow q}$ be a vector of $m \geq 0$ declarations $(X_i : \Theta_i \Rightarrow q_i)$ where no fixed-point variable name and no sequent occurs twice. The specification of $\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \Xi)$ is as follows:*

If, for some $1 \leq i \leq m$, $p = q_i$ and $\Theta_i \subseteq \Gamma$ and $|\Theta_i| = |\Gamma| \cup \{A_1, \ldots, A_n\}$, then

$$\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \Xi) = \lambda z_1^{A_1} \cdots z_n^{A_n}.X_i^{\sigma} \ ,$$

where i is taken to be the biggest such index. Otherwise,

$$\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \Xi) = \lambda z_1^{A_1} \cdots z_n^{A_n}.\mathsf{gfp}\ Y^{\sigma}. \sum_{(y:\vec{B} \supset p) \in \Delta} y\langle \mathcal{F}(\Delta \Rightarrow B_j; \Xi, Y : \sigma) \rangle_j$$

where, in both cases, $\Delta := \Gamma, z_1 : A_1, \ldots, z_n : A_n$ and $\sigma := \Delta \Rightarrow p$ (again the convention on bound variables guarantees that Δ is a context). In the latter case, Y is tacitly supposed not to occur in Ξ (otherwise, the extended list of declarations would not be well-formed).

Notice that, in the first case, the leading λ-abstractions bind variables in the type superscript σ of X_i, and that the condition $\Theta_i \subseteq \Gamma$—and not $\Theta_i \subseteq \Delta$—underlines that the fresh variables cannot be consulted although their types enter well into the next condition $|\Theta_i| = |\Gamma| \cup \{A_1, \ldots, A_n\}$, which is equivalent to $|\Theta_i| = |\Delta|$ (of which only $|\Theta_i| \supseteq |\Delta|$ needs to be checked). The first case represents the situation when the solution space is already captured by a purported solution X_i for the sequent $\Theta_i \Rightarrow p$ with the proper target atom, with all hypotheses in Θ_i available in Γ and, finally, no more formulas available for proof search in the extended current context Δ than in Θ_i. Hence, the purported solution X_i only needs to be expanded by co-contraction in order to cover the solution space for σ (as will be confirmed by Theorem 54). Ambiguity in the choice of i will never appear when starting with the empty vector of declarations (as seen in the proof of the next lemma). The second case translates the semantic definition of solution spaces (Definition 4) into syntax, where the hidden circularity (that is semantically justified by the coinductive reading of Definition 4) is now explicit in terms of the gfp operator that binds the fixed-point variable that is typed according to the $(m+1)$th declaration. The extended list of declarations still does not have a sequent twice: if σ occurred in Ξ, then the first case of the definition would have applied (by freshness of the vector of z_i w. r. t. Ξ, one would know that $n = 0$ in the definition).[8]

In the sequel, we will omit the second argument Ξ to $\mathcal{F}$ in case Ξ is the empty vector of declarations ($m = 0$ in the definition).

Note that, whenever one of the sides of the following equation is defined, then so is the other, and the equation holds (it is important to use variables z_i that are "fresh" w. r. t. Ξ):

$$\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \Xi) = \lambda z_1^{A_1} \cdots z_n^{A_n}.\mathcal{F}(\Gamma, z_1 : A_1, \ldots, z_n : A_n \Rightarrow p; \Xi)$$

[8] Ambiguity in the first clause could be avoided from the outset if Ξ was assumed not only not to have repeated sequents but even no two sequents whose strippings are equal—see right before Lemma 52—but this would not be an invariant in the recursive case since the case analysis is not only driven by the stripped sequents but also by inclusion of one of the sequents in Γ.

Figure 6: Steps towards calculating $\mathcal{F}(\Rightarrow \mathsf{DNPEIRCE})$

$$
\begin{array}{rcl}
\mathcal{F}(\Rightarrow A) & = & \lambda x^{A_0 \supset q}.N_1' \\
N_1' & = & \mathsf{gfp}\ X_1^{x \Rightarrow q}.x\langle \mathcal{F}(x \Rightarrow A_0; X_1)\rangle \\
\mathcal{F}(x \Rightarrow A_0; X_1) & = & \lambda y^{(p\supset q)\supset p}.N_3' \\
N_3' & = & \mathsf{gfp}\ X_2^{x,y \Rightarrow p}.y\langle \mathcal{F}(x, y \Rightarrow p \supset q; X_1, X_2)\rangle \\
\mathcal{F}(x, y \Rightarrow p \supset q; X_1, X_2) & = & \lambda z^p.N_5' \\
N_5' & = & \mathsf{gfp}\ X_3^{x,y,z \Rightarrow q}.x\langle \mathcal{F}(x, y, z \Rightarrow A_0; X_1, X_2, X_3)\rangle \\
\mathcal{F}(x, y, z \Rightarrow A_0; X_1, X_2, X_3) & = & \lambda y_1^{(p\supset q)\supset p}.N_7' \\
N_7' & = & \mathsf{gfp}\ X_4^{x,y,z,y_1 \Rightarrow p}. \\
 & & y\langle \mathcal{F}(x, y, z, y_1 \Rightarrow p \supset q; X_1, X_2, X_3, X_4)\rangle + z + \\
 & & y_1\langle \mathcal{F}(x, y, z, y_1 \Rightarrow p \supset q; X_1, X_2, X_3, X_4)\rangle \\
\mathcal{F}(x, y, z, y_1 \Rightarrow p \supset q; X_1, X_2, X_3, X_4) & = & \lambda z_1^p.N_9' \\
N_9' & = & X_3^{x,y,z,y_1,z_1 \Rightarrow q}
\end{array}
$$

Example 50 (Examples 10 and 35 continued) *We calculate the finitary term representing the solution space for the twice negated Peirce formula $A := \mathsf{DNPEIRCE}$, writing A_0 for PEIRCE. The successive steps are seen in Fig. 6 where we continue with the omission of formulas in the left-hand sides of sequents. For brevity, we do not repeat the sequents associated with the fixed-point variables. The names of intermediary terms are chosen for easy comparison with Example 10. The fixed-point variables X_1, X_2 and X_4 thus have no occurrences in $\mathcal{F}(\Rightarrow A)$, and, as announced before, we will omit them in our resulting finitary term*

$$\mathcal{F}(\Rightarrow \mathsf{DNPEIRCE}) = \lambda x^{\mathsf{PEIRCE}\supset q}.x\langle \lambda y^{(p\supset q)\supset p}.y\langle \lambda z^p.N_5'\rangle\rangle$$

with

$$N_5' = \mathsf{gfp}\ X_3^{x,y,z\Rightarrow q}.x\langle \lambda y_1^{(p\supset q)\supset p}.y\langle \lambda z_1^p.X_3^{x,y,z,y_1,z_1\Rightarrow q}\rangle + z + y_1\langle \lambda z_1^p.X_3^{x,y,z,y_1,z_1\Rightarrow q}\rangle\rangle \ ,$$

still omitting the formulas in the left-hand sides of the sequents.

Example 51 *For the other examples, we have the following representations.*

- $\mathcal{F}(\mathsf{BOOLE}) = \lambda x^p.\lambda y^p.x + y$.
- $\mathcal{F}(\mathsf{INFTY}) = \lambda f^{p\supset p}.\mathsf{gfp}\ X^{f:p\supset p\Rightarrow p}.f\langle X^{f:p\supset p\Rightarrow p}\rangle$.
- $\mathcal{F}(\mathsf{CHURCH}) = \lambda f^{p\supset p}.\lambda x^p.\mathsf{gfp}\ X^\sigma.f\langle X^\sigma\rangle + x$ *with* $\sigma := f : p \supset p, x : p \Rightarrow p$.
- $\mathcal{F}(\mathsf{PEIRCE}) = \lambda x^{(p\supset q)\supset p}.x\langle \lambda y^p.\mathbb{O}\rangle$ *(using $\mathbb{O}$ for the empty sum under the omitted* gfp*).*
- $\mathcal{F}(\mathsf{THREE}) = \lambda x^{(p\supset p)\supset p}.x\langle \lambda y^p.\mathsf{gfp}\, Y^{\sigma_1}.x\langle \lambda z^p.Y^{\sigma_2}\rangle + y\rangle$ *with* $\sigma_1 := x : (p \supset p) \supset p, y : p \Rightarrow p$, $\sigma_2 := x : (p \supset p) \supset p, y : p, z : p \Rightarrow p$, *hence* $\sigma_1 \leq \sigma_2$.

Notice that for INFTY, CHURCH *and* THREE, *the presentation of the solution spaces had already been brought close to this format thanks to cycle analysis that guided the unfolding process, and Thm. 54 below ensures that this works for any sequent.*

Strictly speaking, Definition 49 is no definition since the recursive calls are not guaranteed to terminate. The following lemma spells out the measure that is recursively decreasing in the definition of $\mathcal{F}(\Gamma \Rightarrow C; \Xi)$ and gives a termination criterion that at least guarantees the existence of $\mathcal{F}(\Gamma \Rightarrow C)$.

To this end, we introduce some definitions. Given $\mathcal{A}$ a finite set of formulas

$$\mathcal{A}^{sub} := \{B \mid \text{there exists } A \in \mathcal{A} \text{ such that } B \text{ is subformula of } A\} \ .$$

We say $\mathcal{A}$ is *subformula-closed* if $\mathcal{A}^{sub} = \mathcal{A}$. A *stripped sequent* is a pair $(\mathcal{B}, p)$, where $\mathcal{B}$ is a finite set of formulas. If $\sigma = \Gamma \Rightarrow p$, then its *stripping* $|\sigma|$ denotes the stripped sequent $(|\Gamma|, p)$. We say $(\mathcal{B}, p)$ *is over* $\mathcal{A}$ if $\mathcal{B} \subseteq \mathcal{A}$ and $p \in \mathcal{A}$. There are $size(\mathcal{A}) := a \cdot 2^k$ stripped sequents over $\mathcal{A}$, if a (resp. k) is the number of atoms (resp. formulas) in $\mathcal{A}$.

Let $\mathcal{A}$ be subformula-closed. We say $\Gamma \Rightarrow C$ and $\Xi := \overrightarrow{X : \Theta \Rightarrow q}$ satisfy the $\mathcal{A}$-*invariant* if:

(i) $|\Gamma| \cup \{C\} \subseteq \mathcal{A}$;

(ii) $\Theta_1 \subseteq \Theta_2 \subseteq \cdots \subseteq \Theta_m = \Gamma$ (if $m = 0$ then this is meant to be vacuously true);

(iii) For $1 \leq j \leq m$, $q_j \in |\Gamma|^{sub}$;

(iv) $size(\Xi) = m$,

where $m \geq 0$ is the length of vector Ξ and $size(\Xi)$ is the number of elements of $|\Xi|$ and $|\Xi| := \{|\sigma| : \sigma \in \Xi\}$ (if $m = 0$, also items (iii) and (iv) are trivially true). (iv) is equivalent to saying that the stripped sequents $|\sigma|$ for $\sigma \in \Xi$ are pairwise different. Notice that this strengthens the global assumption that no sequent occurs twice in Ξ. In particular, there is no more ambiguity in the choice of an i in the first case of Definition 49—the choice of the biggest such i there is only to ensure definiteness.

If the $\mathcal{A}$-invariant is satisfied, then $|\sigma|$ is over $\mathcal{A}$, for all $\sigma \in \Xi$ (*).[9]

Lemma 52 *If σ and Ξ satisfy the $\mathcal{A}$-invariant, for some $\mathcal{A}$ subformula-closed, then $\mathcal{F}(\sigma; \Xi)$ is well-defined. In particular, for all sequents σ, $\mathcal{F}(\sigma)$ is well-defined.*

Proof As in the definition, we consider a sequent of the form $\Gamma \Rightarrow C$ with $C = \vec{A} \supset p$. Let us call *recursive call* a "reduction"

$$\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \overrightarrow{X : \Theta \Rightarrow q}) \rightsquigarrow \mathcal{F}(\Delta \Rightarrow B_j; \overrightarrow{X : \Theta \Rightarrow q}, Y : \sigma) \tag{17}$$

where the if-guard in Def. 49 fails; Δ and σ are defined as in the same definition; and, for some y, $(y : \vec{B} \supset p) \in \Delta$. We want to prove that every sequence of recursive calls from $\mathcal{F}(\Gamma \Rightarrow C)$ is finite.

We prove that, if $\Gamma \Rightarrow C$ and Ξ satisfy the $\mathcal{A}$-invariant for some subformula-closed $\mathcal{A}$, then every sequence of recursive calls from $\mathcal{F}(\Gamma \Rightarrow C; \Xi)$ is finite. The proof is by induction on $size(\mathcal{A}) - m$ which is non-negative thanks to (iv) and observation (*) above ($|\Xi|$ represents some of the stripped sequents over $\mathcal{A}$).

Let $C = \vec{A} \supset p$. We analyze an arbitrary recursive call (17) and prove that every sequence of recursive calls from $\mathcal{F}(\Delta \Rightarrow B_j; \Xi, Y : \sigma)$ is finite. This is achieved by proving that $\Delta \Rightarrow B_j$ and $\Xi, Y : \sigma$ satisfy the $\mathcal{A}$-invariant since $size(\mathcal{A}) - (m + 1) < size(\mathcal{A}) - m$ then allows to use the inductive hypothesis.

By assumption, (i), (ii), (iii) and (iv) above hold. We want to prove:

(i') $|\Delta| \cup \{B_j\} \subseteq \mathcal{A}$;

(ii') $\Theta_1 \subseteq \Theta_2 \subseteq \cdots \subseteq \Theta_m \subseteq \Delta = \Delta$;

(iii') For $1 \leq j \leq m + 1$, $q_j \in |\Delta|^{sub}$;

(iv') $size(\Xi, Y : \sigma) = m + 1$.

Proof of (i'). $|\Delta| = |\Gamma| \cup \{A_1, \ldots, A_n\} \subseteq \mathcal{A}$ by (i) and $\mathcal{A}$ subformula-closed. B_j is a subformula of $\vec{B} \supset p$ and $\vec{B} \supset p \in |\Delta|$ because $(y : \vec{B} \supset p) \in \Delta$, for some y.

Proof of (ii'). Immediate by (ii) and $\Gamma \subseteq \Delta$.

Proof of (iii'). For $1 \leq j \leq m$, $q_j \in |\Gamma|^{sub} \subseteq |\Delta|^{sub}$, by (iii) and $\Gamma \subseteq \Delta$. On the other hand, $q_{j+1} = p \in |\Delta|^{sub}$ because $(y : \vec{B} \supset p) \in \Delta$, for some y.

[9] This would be even more direct with the following relaxation of (iii): For $1 \leq j \leq m$, $q_j \in \mathcal{A}$. This latter condition could effectively replace (iii) in the definition of $\mathcal{A}$-invariant for the purposes of our proofs.

Proof of (iv'). Given that the if-guard of Def. 49 fails, and that $\Theta_i \subseteq \Gamma$ due to (ii), we conclude: for all $1 \leq i \leq m$, $p \neq q_i$ or $|\Theta_i| \neq |\Delta|$. But this means that $|\sigma| = |\Delta \Rightarrow p| \notin |\Xi|$, hence $\mathit{size}(\Xi, Y : \sigma) = \mathit{size}(\Xi) + 1 = m + 1$ by (iv).

Finally, to justify the particular case, let $\mathcal{A} = (|\Gamma| \cup \{C\})^{sub}$ and observe that $\Gamma \Rightarrow C$ and the empty vector of declarations satisfy the $\mathcal{A}$-invariant. □

To conclude, we have justified the definition of $\mathcal{F}(\sigma)$ for all sequents σ, but we allow ourselves to write $\mathcal{F}(\sigma; \Xi)$ also in cases that are not covered by the previous proof. It will be understood that this is meant to be under the proviso of definedness.

The main objective of the typing system in Section 5.2 is obtained by the following result:

Lemma 53 (Finitary representation is well-typed, hence well-bound) *If $\mathcal{F}(\Gamma \Rightarrow C; \Xi)$ is defined, we have*

$$\Xi \rfloor \Gamma \vdash \mathcal{F}(\Gamma \Rightarrow C; \Xi) : C \ .$$

In particular, $\Gamma \vdash \mathcal{F}(\Gamma \Rightarrow C) : C$.

Proof By structural recursion on the obtained finitary term $\mathcal{F}(\Gamma \Rightarrow C; \Xi)$. Notice that the context weakening built into the gfp rule in Fig. 5 is not needed for this result (i. e., Θ and Γ of that rule can always agree). □

5.4 Equivalence of representations

Now, we establish the result on the equivalence of the coinductive and inductive representations of the solution spaces. For this, we need the coarser equivalence relation $=$ on Böhm forests because of the rather rough way co-contraction operates that takes identification up to symmetry and idempotence of the sum operation for the elimination alternatives for granted. The proof below is a revision of the proof of [EMP13, Theorem 24] in the light of the new notion of environments and their admissibility w. r. t. a term.

Theorem 54 (Equivalence) *For any sequent σ, there exists $\mathcal{F}(\sigma) \in \overline{\lambda}_{\Sigma}^{\mathsf{gfp}}$ with no free occurrences of fixed-point variables such that $[\![\mathcal{F}(\sigma)]\!] = \mathcal{S}(\sigma)$.*

Proof

For a vector $\Xi = \overrightarrow{X : \Theta \Rightarrow q}$ satisfying the requirements in Definition 49, the mapping ξ_Ξ obtained by setting $\xi_\Xi(X_i^{\Theta_i \Rightarrow q_i}) := \mathcal{S}(\Theta_i \Rightarrow q_i)$ is an environment.

We prove that, if $\mathcal{F}(\sigma; \Xi)$ is well-defined, then

(i) for every $X^{\sigma'} \in \mathit{FPV}(\mathcal{F}(\sigma; \Xi))$, there is $X : \sigma'' \in \Xi$ such that $\sigma'' \leq \sigma'$, hence ξ_Ξ is admissible for $\mathcal{F}(\sigma; \Xi)$;

(ii) $\mathcal{F}(\sigma; \Xi)$ is well-bound;

(iii) $[\![\mathcal{F}(\sigma; \Xi)]\!]_{\xi_\Xi} = \mathcal{S}(\sigma)$.

The theorem follows by taking for Ξ the empty vector, since, by Lemma 52, $\mathcal{F}(\sigma)$ is well-defined. However, the properties (i), (ii) and (iii) hold whenever $\mathcal{F}(\sigma; \Xi)$ exists.

The proof is by structural induction on the term $\mathcal{F}(\sigma; \Xi)$.

Let $\sigma = \Gamma \Rightarrow \vec{A} \supset p$ and $\Delta := \Gamma, z_1 : A_1, \ldots, z_n : A_n$, as in Definition 49.

Case $p = q_i$ and $\Theta_i \subseteq \Gamma$ and $|\Theta_i| = |\Delta|$, for some $1 \leq i \leq m$, which implies $(\Theta_i \Rightarrow q_i) \leq (\Delta \Rightarrow p)$ (*).

(i) The unique fixed-point variable in $\mathcal{F}(\sigma; \Xi)$ is $X_i^{\Delta \Rightarrow p}$, and we observe that $\Theta_i \Rightarrow q_i \leq \Delta \Rightarrow p$ and $X_i : \Theta_i \Rightarrow q_i \in \Xi$.

(ii) There is no occurrence of gfp in $\mathcal{F}(\sigma; \Xi)$.

(iii)

$$\begin{array}{rcll} LHS & = & \lambda z_1^{A_1} \cdots z_n^{A_n}.[\![X_i^{\Delta \Rightarrow p}]\!]_{\xi_\Xi} & \text{(by definition)} \\ & = & \lambda z_1^{A_1} \cdots z_n^{A_n}.[\Delta \Rightarrow p/\Theta_i \Rightarrow q_i]\xi_\Xi(X_i^{\Theta_i \Rightarrow q_i}) & \text{(by definition and (*) above)} \\ & = & \lambda z_1^{A_1} \cdots z_n^{A_n}.[\Delta \Rightarrow p/\Theta_i \Rightarrow q_i]\mathcal{S}(\Theta_i \Rightarrow q_i) & \text{(by definition of } \xi_\Xi) \\ & = & \lambda z_1^{A_1} \cdots z_n^{A_n}.\mathcal{S}(\Delta \Rightarrow p) & \text{(by Lemma 33 and (*))} \\ & = & RHS & \text{(by definition)} \end{array}$$

The inductive case is essentially an extension of the inductive case in [EMP13, Theorem 15] for the Horn fragment. Suppose the case above holds for no $1 \leq i \leq m$.

(i) Follows from part (i) of the inductive hypothesis (recall that a binder $\mathsf{gfp}\, Y^{\sigma'}$ binds all occurrences of $Y^{\sigma''}$).

(ii) Follows from parts (ii) and (i) of the inductive hypothesis for the inner occurrences of gfp and the outermost occurrence, respectively.

(iii) $LHS = \lambda z_1^{A_1} \cdots z_n^{A_n}.N^\infty$, where N^∞ is the unique solution of the following equation

$$N^\infty \quad = \quad \sum_{(y:\vec{B} \supset p) \in \Delta} y\langle [\![\mathcal{F}(\Delta \Rightarrow B_j; \Xi, Y : \sigma')]\!]_{\xi_\Xi \cup [Y^{\sigma'} \mapsto N^\infty]} \rangle_j \tag{18}$$

where $\sigma' := \Delta \Rightarrow p$. Now observe that, by inductive hypothesis, the following equations (19) and (20) are equivalent.

$$\mathcal{S}(\sigma') \quad = \quad \sum_{(y:\vec{B} \supset p) \in \Delta} y\langle [\![\mathcal{F}(\Delta \Rightarrow B_j; \Xi, Y : \sigma')]\!]_{\xi_{(\Xi, Y:\sigma')}} \rangle_j \tag{19}$$

$$\mathcal{S}(\sigma') \quad = \quad \sum_{(y:\vec{B} \supset p) \in \Delta} y\langle \mathcal{S}(\Delta \Rightarrow B_j) \rangle_j \tag{20}$$

By definition of $\mathcal{S}(\sigma')$, (20) holds (even w. r. t. $=$); hence, since $\xi_{(\Xi, Y:\sigma')} = \xi_\Xi \cup [Y^{\sigma'} \mapsto \mathcal{S}(\sigma')]$ and because of (19), $\mathcal{S}(\sigma')$ is the solution N^∞ of (18) modulo $=$. Therefore $LHS = \lambda z_1^{A_1} \cdots z_n^{A_n}.\mathcal{S}(\sigma')$, and the latter is RHS by definition of $\mathcal{S}(\Gamma \Rightarrow \vec{A} \supset p)$. □

Corollary 55 *$\mathcal{F}(\sigma; \Xi)$ is regular.*

Proof By Lemma 39, $\mathcal{F}(\sigma; \Xi)$ is regular since ξ_Ξ in the proof above is admissible for it. □

See the technical Appendix A for an even stronger result than regularity.

Corollary 56 *For every $M \in \overline{\lambda}^{co}$, $\mathsf{mem}(M, [\![\mathcal{F}(\sigma)]\!])$ iff $\mathsf{mem}(M, \mathcal{S}(\sigma))$.*

Proof Obviously, membership is not affected by bisimilarity $=$, and by our extension to $=$ neither. □

The equivalence theorem may be seen as achieving completeness for the finitary representation of solution spaces: every solution space is the semantics of some finitary term. Such completeness cannot be expected at the level of individual solutions. Take, for instance, $\Gamma = x_0 : p \supset p, \ldots, x_9 : p \supset p$. Then $\mathcal{S}(\Gamma \Rightarrow p)$ is the Böhm forest N such that $N = x_0 < N > + \cdots + x_9 < N >$, one of whose members is, say, the decimal expansion of π.

Although solution spaces may have irrational members, they have "rationality" as a collection, since essentially—not taking into account contraction phenomena—they are generated by repeating infinitely a choice from a fixed menu. It is this "rationality" that can be expressed by finitary terms.

6 Final remarks

Contribution. We are developing a comprehensive approach to reductive proof search that is naturally integrated with the Curry-Howard isomorphism: the λ-terms used to represent proofs are seen co-inductively in order to capture (possibly infinite) solutions of search problems. But this Curry-Howard representation is just a convenient definition of the structures generated by proof search. An effective analysis has to be conducted in an accompanying, equivalent, finitary representation, which may be seen as the main technical contribution. The role of formal sums also stands out, specially in connection with the new operation of co-contraction. Finally, the design of the finitary calculus is noteworthy, with its combination of formal sums, fixed points, and a relaxed form of fixed-point variable binding, capable of cycle detection through the type system.

Surely other case studies are needed in order to test the comprehensiveness of the approach, although it is easy to anticipate that our main theorem, about the equivalence of representations, rests on the subformula property of the object logic. We preferred to explore a simple case study (proof search in LJT) in order separate the complexities of the proposed approach for proof search from the complexities of the object logic. In future work (see below) we plan to explore further the approach over the same case study, before moving to richer logics. On the other hand, given the bijection between our $\overline{\lambda}$-terms and beta-normal, eta-long lambda-terms, and given the Curry-Howard isomorphism, search problems in LJT correspond to inhabitation problems in the simply-typed lambda-calculus, and so, for the study of the latter kind of problems, our level of generality may already prove useful.

Related work. In the context of logic programming with classical first-order Horn clauses, the use of co-inductive structures is seen in [KP11], in order to provide a uniform algebraic semantics for both finite and infinite SLD-resolutions. In [PR04] we find a comprehensive approach to proof search, where the generalization of proofs to searches (or "reductions") is accounted for semantically. Parigot's $\lambda\mu$-calculus is used to represent proofs in classical and intuitionistic sequent calculus, but no indication is given on how such terms could represent searches.

In Sect. 1.3.8 of [BDS13] we find a list of types, for each of which the set of inhabitants is described through an "inhabitation machine". This list covers among others all our examples in Ex. 1 with the exception of INFTY and DNPEIRCE. We invite the reader to compare those descriptions in graphical presentation in the cited book with our succinct presentations of the solution spaces worked out in Sect. 3 and Sect. 4 (see Exs. 6, 8, 14, and 36). While our expressions do not display the types of the subexpressions, they are explicit about when a variable gets available for binding (in their example (vii), their variable x, that corresponds to our y in Ex. 36, looks as if it was available from the outset), and our expressions are even more explicit about the generation process for new names (the book speaks about "new incarnations") using standard lambda abstractions and the co-contraction operator.

While our presentations of solution spaces in Sect. 3 and Sect. 4 are still on the informal level of infinitary terms with meta-level fixed points, and for that reason may seem far from a "machine" for the generation of inhabitants, the finitary expressions we obtained in Ex. 50 and Ex. 51 with the machinery of Sect. 5 compare in the same way with the inhabitation machines of [BDS13] and are proper syntactic elements and can thus qualify as "machine" descriptions of the process of obtaining the inhabitants (and even the infinite solutions—notice that infinite solutions are not addressed at all in the description of inhabitation machines in [BDS13]).

The work [SDB15] also studies mathematical structures for representing proof search, and can partly be seen as offering a realisation of the intuitive description of the inhabitation machines in [BDS13]. Similarly to our work, [SDB15] handles search for normal inhabitants in the simply-typed lambda-calculus. However, the methods of [SDB15] are very different from ours. Their methods come from automata and language theory, and proof search is represented through automata with a non-standard form of register, as a way to avoid automata with infinite alphabets, and still meet the need for a supply of infinitely many bound variables in types like DNPEIRCE or the monster type. Again, unlike in our work, infinite solutions are not a concern of [SDB15].

Only seemingly related work. Logics with fixed points or inductive definitions [San02, BS11] admit infinite or "circular" proofs, which are infinite "pre-proofs" enjoying an extra global, semantic condition to ensure that only valid conclusions are allowed. In addition, the proofs of these logics have alternative (sometimes equivalent) finite representations as graphs with cycles (e.g. trees with back-edges). Despite superficial similarity, bear in mind the several differences relatively to what is done in the present paper: first, there is the conceptual difference between solution and proof; second, in our simple object logic, proofs are the finite solutions (hence trivially filtered amongst solutions), and therefore infinite solutions never correspond to globally correct reasoning; third, fixed points are not present in the object logic, but rather in the finitary calculus, which works, at best, as a meta-logic.

Future work. We would like to profit from the finitary representation of a solution space to extract individual solutions. As suggested in Section 2.2, this can be done by pruning the solution space, but unfolding of fixed points will be involved in the extraction process as well. This is a base for the accounting of algorithmic control in proof search through rewriting.

In companion work in preparation, we are using the finitary system $\overline{\lambda}_{\Sigma}^{\mathsf{gfp}}$, and the effective representation of solution spaces of proof search problems as terms of this system provided by the mapping $\mathcal{F}$, to develop an approach to the study of inhabitation problems in the simply-typed lambda-calculus, by which we mean both decision problems, like "does type A have an inhabitant?" or "does type A have finitely many inhabitants", and related questions like counting or listing the inhabitants of a type known to have finitely many of them [Hin97]. In order to test for the generality of our coinductive approach to proof search, in particular, we intend to extend it to treat the first-order case. Recall that first-order Horn logic receives a coalgebraic semantics in [KP11]. Success along this path could provide a basis for a coinductive extension of λ-Prolog programming with first-order hereditary Harrop formulas [MN12], where the possibility of negative occurrences of implication raises the need for dealing with programs to which clauses may be added dynamically.

Acknowledgements. Ralph Matthes is funded by the *Climt* project (ANR-11-BS02-016 of the French Agence Nationale de la Recherche). José Espírito Santo and Luís Pinto are both funded by Portuguese Funds through FCT – Fundação para a Ciência e a Tecnologia, within the Project UID/MAT/00013/2013.

References

[BDS13] Henk Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in Logic. Cambridge University Press, 2013.

[BS11] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21(6):1177–1216, 2011.

[DP99] Roy Dyckhoff and Luís Pinto. Proof search in constructive logics. In S.B. Cooper and J. K. Truss, editors, *Sets and Proofs: invited papers from Logic Colloquium'97*, pages 53–65, 1999.

[EMP13] José Espírito Santo, Ralph Matthes, and Luís Pinto. A coinductive approach to proof search. In David Baelde and Arnaud Carayol, editors, *Proceedings of FICS 2013*, volume 126 of *EPTCS*, pages 28–43, 2013.

[Her95] Hugo Herbelin. A λ-calculus structure isomorphic to a Gentzen-style sequent calculus structure. In L. Pacholski and J. Tiuryn, editors, *Proceedings of CSL'94*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1995.

[Hin97] J. Roger Hindley. *Basic Simple Type Theory*, volume 42 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1997.

[Joa04] Felix Joachimski. Confluence of the coinductive λ-calculus. *Theor. Comput. Sci.*, 311(1-3):105–119, 2004.

[KP11] Ekaterina Komendantskaya and John Power. Coalgebraic derivations in logic programming. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*, pages 352–366. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[LM09] Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logic. *Theoretical Computer Science*, 410:4747–4768, 2009.

[MN12] Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, June 2012.

[NUB11] Keiko Nakata, Tarmo Uustalu, and Marc Bezem. A proof pearl with the fan theorem and bar induction - walking through infinite trees with mixed induction and coinduction. In Hongseok Yang, editor, *APLAS*, volume 7078 of *LNCS*, pages 353–368. Springer, 2011.

[PM12] Celia Picard and Ralph Matthes. Permutations in coinductive graph representation. In Dirk Pattinson and Lutz Schröder, editors, *Coalgebraic Methods in Computer Science (CMCS 2012)*, volume 7399 of *Lecture Notes in Computer Science, IFIP subseries*, pages 218–237. Springer, 2012.

[PR04] David J. Pym and Eike Ritter. *Reductive Logic and Proof-search: Proof Theory, Semantics, and Control*. Oxford Logic Guides. Oxford University Press, 2004.

[San02] Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In M. Nielsen and U. Engberg, editors, *Foundations of Software Science and Computation Structures (FOSSACS 2002), Proceedings*, volume 2303 of *LNCS*, pages 357–371. Springer, 2002.

[SDB15] Aleksy Schubert, Wil Dekkers, and Henk P. Barendregt. Automata theoretic account of proof search. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, volume 41 of *LIPIcs*, pages 128–143. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

A Technical Appendix on Regularity of Finitary Terms

In Section 5, we insisted that we do not confine our investigation to trivially regular terms. This is directly imposed by Definition 49, as we will see next.

Example 57 (A not trivially regular term) *Assume three different atoms p, q, r, set $\Gamma := y_1 : q \supset p, y_2 : (r \supset q) \supset p, x : r$ and $\Xi := X : \Gamma \Rightarrow q$. Then Definition 49 yields*

$$\mathcal{F}(\Gamma \Rightarrow p; \Xi) = \mathsf{gfp}\, Y^{\Gamma \Rightarrow p}.y_1\langle X^{\Gamma \Rightarrow q}\rangle + y_2\langle \lambda z^r.X^{\Gamma, z:r \Rightarrow q}\rangle$$

Fixed-point variable X occurs free in this expression with two different sequents as types, hence the expression is not trivially regular.

Definition 49 even leads us to consider trivially regular terms with regular but not trivially regular subterms, hidden under a greatest fixed-point construction:

Example 58 (Hidden irregularity) *Consider the following modification of the previous example: add the binding $y : p \supset q$ to Γ. Then, the above calculation of $\mathcal{F}(\Gamma \Rightarrow p; \Xi)$ comes to the same result. And we calculate*

$$\mathcal{F}(\Gamma \Rightarrow q) = \mathsf{gfp}\, X^{\Gamma \Rightarrow q}.y\langle \mathcal{F}(\Gamma \Rightarrow p; \Xi)\rangle$$

Hence, X with two different sequents as types has to be bound by the outer fixed-point operator.

The following notion may be of further use:

Definition 59 (strongly regularity in $\overline{\lambda}_{\Sigma}^{\mathsf{gfp}}$) *An expression T in $\overline{\lambda}_{\Sigma}^{\mathsf{gfp}}$ is strongly regular, if all subexpressions of T (including T) are regular.*

We can even strengthen Corollary 55.

Corollary 60 *$\mathcal{F}(\Gamma \Rightarrow C; \Xi)$ is strongly regular (whenever it exists).*

Proof Regularity is already expressed in Corollary 55. Concerning the regularity of the subexpressions, λ-abstraction does not influence on regularity, and in the recursive case of the definition of $\mathcal{F}(\Gamma \Rightarrow C; \Xi)$, the same $\xi_{\Xi, Y:\sigma}$ is admissible for all the occurring subterms, hence also for the summands that are bound by the gfp operation. □