Some aspects of coinductive proof search in intuitionistic propositional logic

Ralph Matthes

CNRS, Inst. Recherche Informatique Toulouse (IRIT), Univ. Toulouse, France

18th Workshop on Proof, Computation, Complexity
PCC 2019

July 15, 2019
Institut Mittag-Leffler, Djursholm (near Stockholm), Sweden

Abstract

In joint work with Espírito Santo and Pinto, we have developed a new representation of the search space for locally correct applications of the proof rules for the cut-free fragment of intuitionistic propositional logic, not limiting the representation to the construction of (finite) inhabitants. In the present talk, I'll focus on

- an analysis of recursion depth in the construction of finitary representations of search spaces (joint work to be submitted),
- "König's lemma" for proof search obtained through pruning of the search space (joint work to appear in journal), and, time permitting,
- a proposal for an extension to full intutionistic propositional logic, hence with conjunction and disjunction (new material by the author).

Overview of the approach

- new approach to study inhabitation-related problems in STLC such as
  - is type A inhabited?
  - is the set of inhabitants of A finite?
  - are all "solutions" of A finite?

Overview of the approach

- new approach to study inhabitation-related problems in STLC such as
  - is type A inhabited?
  - is the set of inhabitants of A finite?
  - are all "solutions" of A finite?
- approach based on a λ-calculus for representing proof search in minimal implicational logic (derived from a coinductive λ-calculus);

Overview of the approach

- new approach to study inhabitation-related problems in STLC such as
  - is type A inhabited?
  - is the set of inhabitants of A finite?
  - are all "solutions" of A finite?
- approach based on a λ-calculus for representing proof search in minimal implicational logic (derived from a coinductive λ-calculus);
- obtained:
  - syntax-directed decision procedures for inhabitation-related problems;
  - simple recursive counting function for finitely inhabited types;
  - perspicuous new proofs of two known refinements of the well-known coherence theorem for balanced formulas;
  - a result in the spirit of König's lemma for proof search

The approach

The base simply-typed λ-calculus (λ)

- searching for inhabitants of A in context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ corresponds by Curry-Howard to search for proofs of sequent $\Gamma \Rightarrow A$;

The base simply-typed λ-calculus (λ)

- searching for inhabitants of A in context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ corresponds by Curry-Howard to search for proofs of sequent $\Gamma \Rightarrow A$;
- a sequent calculus for minimal implicational logic tailored for proof search:
  - let p range over atoms, and $\vec{A} \supset p$ abbreviate $A_1 \supset \ldots \supset A_n \supset p$ (assumed to be p if $\vec{A}$ is empty);
  - $$\frac{\Gamma, x : A \Rightarrow B}{\Gamma \Rightarrow A \supset B} \qquad \frac{(x : \vec{A} \supset p) \in \Gamma \quad \forall i, \Gamma \Rightarrow A_i}{\Gamma \Rightarrow p}$$

The base simply-typed λ-calculus (λ)

- searching for inhabitants of A in context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ corresponds by Curry-Howard to search for proofs of sequent $\Gamma \Rightarrow A$;
- a sequent calculus for minimal implicational logic tailored for proof search:
  - let p range over atoms, and $\vec{A} \supset p$ abbreviate $A_1 \supset \ldots \supset A_n \supset p$ (assumed to be p if $\vec{A}$ is empty);
  - $$\frac{\Gamma, x{:}A \Rightarrow B}{\Gamma \Rightarrow A \supset B} \qquad \frac{(x : \vec{A} \supset p) \in \Gamma \quad \forall i, \Gamma \Rightarrow A_i}{\Gamma \Rightarrow p}$$
- the corresponding STLC captures exactly the typable η-long β-normal λ-terms, having typing rules:

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

A coinductive λ-calculus for proof search (λ^{co}_{Σ})

- proof search proceeds by bottom-up application of rules, but infinite trees of sequents can be generated, and there may be choice points, e. g., think of type of Church numerals: $(p \supset p) \supset (p \supset p)$;

A coinductive λ-calculus for proof search (λ_{Σ}^{co})

- proof search proceeds by bottom-up application of rules, but infinite trees of sequents can be generated, and there may be choice points, e. g., think of type of Church numerals: $(p \supset p) \supset (p \supset p)$;
- solutions (single runs of proof search) and the whole solution space are naturally represented through a coinductive λ-calculus with sums (but only finite solutions correspond to proofs):

A coinductive λ-calculus for proof search (λ_{Σ}^{co})

- proof search proceeds by bottom-up application of rules, but infinite trees of sequents can be generated, and there may be choice points, e. g., think of type of Church numerals: $(p \supset p) \supset (p \supset p)$;
- solutions (single runs of proof search) and the whole solution space are naturally represented through a coinductive λ-calculus with sums (but only finite solutions correspond to proofs):
  - co-terms with sums (called forests by us):

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

    co-terms in the sense of a coinductive data structure!
  - typing rule for sums:

$$\frac{\forall i, \Gamma \vdash E_i : p}{\Gamma \vdash (E_1 + \cdots + E_n) : p}\ \mathit{Alts}$$

Coinductive representation of solution spaces

- individual inductive or even coinductive lambda-terms can be seen as members of a forest, with a relation $\mathrm{mem}(N, T)$ for N a term and T a forest;
- the forests are our semantics, seen as search spaces;

Coinductive representation of solution spaces

- individual inductive or even coinductive lambda-terms can be seen as members of a forest, with a relation $\mathrm{mem}(N, T)$ for N a term and T a forest;
- the forests are our semantics, seen as search spaces;
- given a sequent $\sigma := \Gamma \Rightarrow \vec{A} \supset p$, we can define the canonical search space $\mathcal{S}(\sigma)$ associated with σ, as a forest: the definition is done corecursively (it is well-defined by guardedness):

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

where $\Delta := \Gamma, \vec{x} : \vec{A}$ – contexts only grow! Choice comes from the tail occurrences in Δ of the target atom p.

Coinductive representation of solution spaces

- individual inductive or even coinductive lambda-terms can be seen as members of a forest, with a relation $\mathrm{mem}(N, T)$ for N a term and T a forest;
- the forests are our semantics, seen as search spaces;
- given a sequent $\sigma := \Gamma \Rightarrow \vec{A} \supset p$, we can define the canonical search space $\mathcal{S}(\sigma)$ associated with σ, as a forest: the definition is done corecursively (it is well-defined by guardedness):

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

  where $\Delta := \Gamma, \vec{x} : \vec{A}$ – contexts only grow! Choice comes from the tail occurrences in Δ of the target atom p.
- the example of Church's type:

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

  (ν meta-level notation for fixed point);

The canonical search space captures the inhabitants

adequacy of the coinductive representation via the coinductive membership relation that has in particular:

$\mathrm{mem}(t, \mathcal{S}(\Gamma \Rightarrow A))$ iff $\Gamma \vdash t : A$ in λ, writing t for inductive terms.

From semantics to syntax

- The forests form a coinductive datatype and are therefore not necessarily finitely described objects. Beware: $\lambda f^{p \supset p}.\lambda x^{p}.\nu N.(f\langle N\rangle + x)$ has a finite description thanks to a meta-level fixed point.

From semantics to syntax

- The forests form a coinductive datatype and are therefore not necessarily finitely described objects. Beware: $\lambda f^{p \supset p}.\lambda x^{p}.\nu N.(f\langle N\rangle + x)$ has a finite description thanks to a meta-level fixed point.
- Need a finitary counterpart to forests. We propose a lambda-calculus with inductively defined terms (called finitary forests) that also has the means of expressing choice points and that comes with a formal fixed-point operator, based on fixed-point variables that are typed with sequents.

From semantics to syntax

- The forests form a coinductive datatype and are therefore not necessarily finitely described objects. Beware: $\lambda f^{p \supset p}.\lambda x^{p}.\nu N.(f\langle N\rangle + x)$ has a finite description thanks to a meta-level fixed point.
- Need a finitary counterpart to forests. We propose a lambda-calculus with inductively defined terms (called finitary forests) that also has the means of expressing choice points and that comes with a formal fixed-point operator, based on fixed-point variables that are typed with sequents.
- There is a natural interpretation of finitary forests as forests, which henceforth allows to specify problems semantically (in terms of forests), to which an effective solution is described in terms of finitary forests.

The λ-calculus for proof search ($\lambda^{\mathrm{gfp}}_{\Sigma}$)

- syntax of expressions (read inductively):

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

  - X ranges over fixed-point variables and sequents σ have atomic RHS, i. e., $\sigma = (\Gamma \Rightarrow p)$;
  - $\mathrm{gfp} X^{\sigma}$ binds all free occurrences of $X^{\sigma'}$ with $\sigma \leq \sigma'$, defined in sloppy words as: σ' may have more declarations than σ, but not with new types;

The λ-calculus for proof search ($\lambda^{\mathrm{gfp}}_{\Sigma}$)

- syntax of expressions (read inductively):

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

  - X ranges over fixed-point variables and sequents σ have atomic RHS, i. e., $\sigma = (\Gamma \Rightarrow p)$;
  - $\mathrm{gfp} X^{\sigma}$ binds all free occurrences of $X^{\sigma'}$ with $\sigma \leq \sigma'$, defined in sloppy words as: σ' may have more declarations than σ, but not with new types;
- semantics of well-bound expressions: T interpreted with the help of environments ξ as forests $[\![T]\!]_{\xi}$; the important clauses:

$$\begin{array}{rcl} [\![X^{\sigma'}]\!]_{\xi} & := & [\sigma'/\sigma]\xi(X^{\sigma}) \quad \text{if } \sigma \leq \sigma' \\ [\![\mathrm{gfp}\, X^{\sigma}.\sum_i E_i]\!]_{\xi} & := & \nu\, N.\sum_i [\![E_i]\!]_{\xi \cup [X^{\sigma} \mapsto N]} \end{array}$$

The operation in the first clause is decontraction (not substitution). The fixed-point construction operates on the meta-level (on forests), too.

Finitary representation of solution spaces

We apply our methodology for the first time:

- The search space for a sequent σ is specified by a forest, namely $\mathcal{S}(\sigma)$.
- We seek a finitary forest whose semantics is just $\mathcal{S}(\sigma)$.
- It will be called the "finitary representation of σ".
- Thus, we get an effective counterpart to the coinductively specified notion.

Finitary representation of solution spaces, concretely

- the finitary representation of the solution space of a sequent σ is a closed and well-bound $\lambda_{\Sigma}^{\mathrm{gfp}}$-term: $\mathcal{F}(\sigma) := \mathcal{F}(\sigma; \emptyset)$.
  $\mathcal{F}(\sigma; \Xi)$ is defined in general: given an appropriate vector Ξ of declarations $(X_i : \Theta_i \Rightarrow q_i)$, $\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \Xi)$ is defined by the following term(s):

  let $\Delta := \Gamma \cup \{z_1 : A_1, \cdots, z_n : A_n\}$ and $\sigma' := \Delta \Rightarrow p$ in

  (1) $\lambda z_1^{A_1} \cdots z_n^{A_n}.X_i^{\sigma'}$, if $p = q_i$, $\Theta_i \subseteq \Gamma$ and $\Theta_i \leq \Delta$, for some i.

  The solution space is "essentially captured" by the provided X_i;

Finitary representation of solution spaces, concretely

- the finitary representation of the solution space of a sequent σ is a closed and well-bound $\lambda^{\mathrm{gfp}}_{\Sigma}$-term: $\mathcal{F}(\sigma) := \mathcal{F}(\sigma; \emptyset)$.
  $\mathcal{F}(\sigma; \Xi)$ is defined in general: given an appropriate vector Ξ of declarations $(X_i : \Theta_i \Rightarrow q_i)$, $\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \Xi)$ is defined by the following term(s):

  let $\Delta := \Gamma \cup \{z_1 : A_1, \cdots, z_n : A_n\}$ and $\sigma' := \Delta \Rightarrow p$ in

  (1) $\lambda z_1^{A_1} \cdots z_n^{A_n}.X_i^{\sigma'}$, if $p = q_i$, $\Theta_i \subseteq \Gamma$ and $\Theta_i \leq \Delta$, for some i.

  The solution space is "essentially captured" by the provided X_i;

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

  A new fixed-point variable has to be "spawned" to represent the search problem that is not yet "essentially captured" by Ξ.

Finitary representation of solution spaces, concretely

- the finitary representation of the solution space of a sequent σ is a closed and well-bound $\lambda^{\mathrm{gfp}}_{\Sigma}$-term: $\mathcal{F}(\sigma) := \mathcal{F}(\sigma; \emptyset)$.
  $\mathcal{F}(\sigma; \Xi)$ is defined in general: given an appropriate vector Ξ of declarations $(X_i : \Theta_i \Rightarrow q_i)$, $\mathcal{F}(\Gamma \Rightarrow \vec{A} \supset p; \Xi)$ is defined by the following term(s):

  let $\Delta := \Gamma \cup \{z_1 : A_1, \cdots, z_n : A_n\}$ and $\sigma' := \Delta \Rightarrow p$ in

  (1) $\lambda z_1^{A_1} \cdots z_n^{A_n}.X_i^{\sigma'}$, if $p = q_i$, $\Theta_i \subseteq \Gamma$ and $\Theta_i \leq \Delta$, for some i.

  The solution space is "essentially captured" by the provided X_i;

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

  A new fixed-point variable has to be "spawned" to represent the search problem that is not yet "essentially captured" by Ξ.

- the example of Church's type (set $\sigma := f : p \supset p, x : p \Rightarrow p$):
  $\mathcal{F}(\Rightarrow (p \supset p) \supset p \supset p) := \lambda f^{p \supset p} \lambda x^{p}.\mathrm{gfp} X^{\sigma}.(f\langle X^{\sigma} \rangle + x)$

Finitary representation is well-defined

The recursive definition need not terminate for arbitrary Ξ, but at least for empty Ξ, hence $\mathcal{F}(\sigma)$ is well-defined. Basically, this exploits the subformula property of (minimal) propositional logic.

Note: argument can be refined along the lines of Alves and Broda (FSCD'18) to get a quadratic bound on the recursion depth—not yet included in a submission for publication.

A bound on the recursion depth

More precisely, the length of all Ξ occurring in the computation of $\mathcal{F}(\Rightarrow A)$ can be bounded by the product of

- the number $|A|^+$ of atoms that have positive occurrences in A and
- the number $|A|^-$ of negative subpremises of A (which are all the A_i in a positive subformula $\vec{A} \supset p$ of A)

Reason: the set of formulas in the hypotheses of the sequents in Ξ increases monotonically. Staying the same means going through positive atoms in the targets. After $|A|^+$ steps, a jump is imposed, but then a negative subpremise of A enters the context. This cannot happen more than $|A|^-$ times.

Positive atoms and negative subpremises are just the right concepts to make suitable invariants based on them hold throughout the recursion.

Finitary representation meets the specification

Equivalence Thm.: for any sequent σ, $[\![\mathcal{F}(\sigma)]\!] = \mathcal{S}(\sigma)$

Note: no need for an environment ξ since there are no free fixed-point variables.

Specifying the inhabitation problem

- Question: is there an inhabitant for a given sequent σ?
- Known to be decidable, even PSPACE-complete (Statman).
- On the (semantic) level of forests, the existence of inhabitants corresponds to the existence of a finite (inductive) member of the forest.
- This existence can be inductively characterized by a predicate exfin on forests:

$$\frac{\mathrm{exfin}(N)}{\mathrm{exfin}(\lambda x^A.N)} \qquad \frac{\mathrm{exfin}(E_j)}{\mathrm{exfin}(\sum_i E_i)} \qquad \frac{\forall i,\ \mathrm{exfin}(N_i)}{\mathrm{exfin}(x\langle N_i\rangle_i)}$$

- By duality between least and greatest fixed points, its complement enjoys a coinductive characterization, by way of a predicate nofin on forests. Good for coinductive reasoning!
- The spec. through forests: $\mathrm{exfin}(\mathcal{S}(\sigma))$ iff σ is inhabited.

Effectively solving the inhabitation problem

- syntax-directed inductive predicate EF_P defined as the counterpart on finitary terms (parameterized by a predicate P on sequents, regarded as the "good" sequents):

$$\frac{P(\sigma)}{\mathrm{EF}_P(X^\sigma)} \qquad \frac{\mathrm{EF}_P(N)}{\mathrm{EF}_P(\lambda x^A.N)} \qquad \frac{\mathrm{EF}_P(E_j)}{\mathrm{EF}_P(\mathrm{gfp} X^\sigma.\sum_i E_i)} \qquad \frac{\forall i,\ \mathrm{EF}_P(N_i)}{\mathrm{EF}_P(x\langle N_i\rangle_i)}$$

  Its negation NEF_P can similarly be given inductively. However, in reality, both are just one recursive definition by recursion over the structure of finitary forests.

Effectively solving the inhabitation problem

- syntax-directed inductive predicate EF_P defined as the counterpart on finitary terms (parameterized by a predicate P on sequents, regarded as the "good" sequents):

$$\frac{P(\sigma)}{\mathrm{EF}_P(X^\sigma)} \qquad \frac{\mathrm{EF}_P(N)}{\mathrm{EF}_P(\lambda x^A.N)} \qquad \frac{\mathrm{EF}_P(E_j)}{\mathrm{EF}_P(\mathrm{gfp} X^\sigma.\sum_i E_i)} \qquad \frac{\forall i,\ \mathrm{EF}_P(N_i)}{\mathrm{EF}_P(x\langle N_i\rangle_i)}$$

  Its negation NEF_P can similarly be given inductively. However, in reality, both are just one recursive definition by recursion over the structure of finitary forests.

- the characterizations are equivalent:
  $\mathrm{EF}_P(T)$ iff $\mathrm{exfin}([\![T]\!])$, for $P \subseteq \mathrm{exfin} \circ \mathcal{S}$ and T of form $\mathcal{F}(\sigma)$.

  The proof needs to speak about arbitary "proper" finitary forests and a variation ("simplification") on $[\![T]\!]$. A crucial step is to witness the predicate nofin through a guarded coinductive process.

Effectively solving the inhabitation problem

- syntax-directed inductive predicate EF_P defined as the counterpart on finitary terms (parameterized by a predicate P on sequents, regarded as the "good" sequents):

$$\frac{P(\sigma)}{\mathrm{EF}_P(X^\sigma)} \qquad \frac{\mathrm{EF}_P(N)}{\mathrm{EF}_P(\lambda x^A.N)} \qquad \frac{\mathrm{EF}_P(E_j)}{\mathrm{EF}_P(\mathrm{gfp} X^\sigma.\sum_i E_i)} \qquad \frac{\forall i,\ \mathrm{EF}_P(N_i)}{\mathrm{EF}_P(x\langle N_i\rangle_i)}$$

  Its negation NEF_P can similarly be given inductively. However, in reality, both are just one recursive definition by recursion over the structure of finitary forests.

- the characterizations are equivalent:
  $\mathrm{EF}_P(T)$ iff $\mathrm{exfin}(\llbracket T \rrbracket)$, for $P \subseteq \mathrm{exfin} \circ \mathcal{S}$ and T of form $\mathcal{F}(\sigma)$.

  The proof needs to speak about arbitary "proper" finitary forests and a variation ("simplification") on $\llbracket T \rrbracket$. A crucial step is to witness the predicate nofin through a guarded coinductive process.

- inhabitation of a sequent σ decided by deciding $\mathrm{EF}_\emptyset(\mathcal{F}(\sigma))$.

Deciding type finiteness

Deciding if a type has only finitely many inhabitants can be done in a similar style (also this problem was known to be PSPACE-complete).

Newer material—already accepted for publiction

- We are now heading for different notions of type finiteness, other than having only finitely many (finite) inhabitants which is the most common notion of type finiteness (we do not change that traditional definition).

Newer material—already accepted for publiction

- We are now heading for different notions of type finiteness, other than having only finitely many (finite) inhabitants which is the most common notion of type finiteness (we do not change that traditional definition).
- As soon as one grants the status of "member" of a sequent σ to any member of $\mathcal{S}(\sigma)$ – including infinite ones – other natural concepts of finiteness are possible. One is the finiteness of any member of σ. This concept may be related to the finiteness of the whole search space, hence a third concept of finiteness, in the spirit of König's lemma.

Newer material—already accepted for publiction

- We are now heading for different notions of type finiteness, other than having only finitely many (finite) inhabitants which is the most common notion of type finiteness (we do not change that traditional definition).
- As soon as one grants the status of "member" of a sequent σ to any member of $\mathcal{S}(\sigma)$ – including infinite ones – other natural concepts of finiteness are possible. One is the finiteness of any member of σ. This concept may be related to the finiteness of the whole search space, hence a third concept of finiteness, in the spirit of König's lemma.
- Our main new result is that all these concepts of finiteness are decidable, and so is the property of absence of members (finite or otherwise), which is an extreme form of unprovability.

Methodology

- For the absence of solutions, we follow the method used for the inhabitation problem.

Methodology

- For the absence of solutions, we follow the method used for the inhabitation problem.
- For the finiteness notions, we abstract away from the specific situation of type finiteness by adding a new parameter to the predicate on finitary forests. Type finiteness is then an instance.

Methodology

- For the absence of solutions, we follow the method used for the inhabitation problem.
- For the finiteness notions, we abstract away from the specific situation of type finiteness by adding a new parameter to the predicate on finitary forests. Type finiteness is then an instance.
- The two newly introduced notions of finiteness are other instances of that parameterized notion.

Methodology

- For the absence of solutions, we follow the method used for the inhabitation problem.
- For the finiteness notions, we abstract away from the specific situation of type finiteness by adding a new parameter to the predicate on finitary forests. Type finiteness is then an instance.
- The two newly introduced notions of finiteness are other instances of that parameterized notion.
- The decision algorithms can be specified and verified in one single proof for the generic finiteness predicate, with a generic predicate on finitary forests.

Absence of solutions is decidable

- On the level of forests, absence of solutions can be characterized inductively by property nosol of forests:

$$\frac{\mathsf{nosol}(N)}{\mathsf{nosol}(\lambda x^{A}.N)} \qquad \frac{\forall i,\ \mathsf{nosol}(E_i)}{\mathsf{nosol}(\sum_i E_i)} \qquad \frac{\mathsf{nosol}(N_j)}{\mathsf{nosol}(x\langle N_i\rangle_i)}$$

Absence of solutions is decidable

- On the level of forests, absence of solutions can be characterized inductively by property nosol of forests:

$$\frac{\mathsf{nosol}(N)}{\mathsf{nosol}(\lambda x^A.N)} \qquad \frac{\forall i,\ \mathsf{nosol}(E_i)}{\mathsf{nosol}(\sum_i E_i)} \qquad \frac{\mathsf{nosol}(N_j)}{\mathsf{nosol}(x\langle N_i\rangle_i)}$$

- Its counterpart on finitary forests is given by

$$\frac{P(\sigma)}{\mathsf{NES_P}(X^\sigma)} \qquad \frac{\mathsf{NES_P}(N)}{\mathsf{NES_P}(\lambda x^A.N)}$$

$$\frac{\forall i,\ \mathsf{NES_P}(E_i)}{\mathsf{NES_P}(\mathsf{gfp}X^\sigma.\sum_i E_i)} \qquad \frac{\mathsf{NES_P}(N_j)}{\mathsf{NES_P}(x\langle N_i\rangle_i)}$$

Absence of solutions is decidable

- On the level of forests, absence of solutions can be characterized inductively by property nosol of forests:

$$\frac{\mathsf{nosol}(N)}{\mathsf{nosol}(\lambda x^A.N)} \qquad \frac{\forall i,\ \mathsf{nosol}(E_i)}{\mathsf{nosol}(\sum_i E_i)} \qquad \frac{\mathsf{nosol}(N_j)}{\mathsf{nosol}(x\langle N_i\rangle_i)}$$

- Its counterpart on finitary forests is given by

$$\frac{P(\sigma)}{\mathsf{NES_P}(X^\sigma)} \qquad \frac{\mathsf{NES_P}(N)}{\mathsf{NES_P}(\lambda x^A.N)}$$

$$\frac{\forall i,\ \mathsf{NES_P}(E_i)}{\mathsf{NES_P}(\mathsf{gfp}X^\sigma.\sum_i E_i)} \qquad \frac{\mathsf{NES_P}(N_j)}{\mathsf{NES_P}(x\langle N_i\rangle_i)}$$

- The statements and proofs are analogous to those for the question of inhabitation.

The generic notion of finiteness for forests

Skipped to save time.

The strongest notion of finiteness we obtain is that a forest is only obtained from the inductive reading of the grammar, in other words, that forest is “finite by definition”.

Solution spaces can be infinite without solutions

$x : p \supset q \supset p \Rightarrow p$ has no solution but an infinite solution space.

We want to have a more compact notion than our generic solution spaces. Problem: the coinductive definition must still be well-formed (guarded).

Pruning the solution space

$\mathrm{ES}_\star$ is defined to be $\neg\mathrm{NES}_{\mathrm{NES}_\emptyset \circ \mathcal{F}}$ and thus decidable.
Extensionally, $\mathrm{ES}_\star \circ \mathcal{F} = \mathrm{exsol} \circ \mathcal{S}$.

Definition (Pruned solution space of a sequent)

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

with $\Delta := \Gamma, \vec{x} : \vec{A}$, where

- $\underline{\lambda}x : A.T := \lambda x : A.T$, if $T \neq \mathbb{O}$; and $\underline{\lambda}x : A.T := \mathbb{O}$, otherwise.
- $(y : \vec{B} \supset p)\underline{\in}\Delta :\Leftrightarrow (y : \vec{B} \supset p) \in \Delta$ and, for all j, $\mathrm{ES}_\star(\mathcal{F}(\Delta \Rightarrow B_j))$.

$\mathbb{O}$ is a notation for the empty sum. We thus avoid lambda-abstractions over an empty sum, and we coinductively avoid summands without solutions.

Analyzing pruning

Lemma

1. *The pruned solution space has the same members as the original one.*
2. *If σ has a solution, the pruned solution space has no empty sum.*
3. *If σ has no solution, the pruned solution space is the empty sum.*

The proof is a bit intricate and is based on coinductive characterizations and then coinductive reasoning.

Part 2 is proved first.

Proof of part 2

Let us write $\mathrm{no}\mathbb{O}(T)$ to mean that T has no empty sum. The conclusion $\mathrm{no}\mathbb{O}(\underline{\mathcal{S}}(\sigma))$ is turned into a coinductive predicate:

$$\frac{\mathrm{no}\mathbb{O}(\underline{\mathcal{S}}(\Gamma, x : A \Rightarrow B))}{\mathrm{no}\mathbb{O}(\underline{\mathcal{S}}(\Gamma \Rightarrow A \supset B))}\ (a)$$

$$\frac{\exists (y : \vec{B} \supset p) \underline{\in} \Gamma \quad \forall (y : \vec{B} \supset p) \underline{\in} \Gamma\ \forall j,\ \mathrm{no}\mathbb{O}(\underline{\mathcal{S}}(\Gamma \Rightarrow B_j))}{\mathrm{no}\mathbb{O}(\underline{\mathcal{S}}(\Gamma \Rightarrow p))}\ (b)$$

Then, the implication can be proven by showing that $\mathrm{exsol}(\mathcal{S}(\sigma))$ is backward closed w. r. t. (a) and (b).

Proof of part 1

We want to prove: $\mathsf{mem}(N, \underline{\mathcal{S}}(\sigma))$ iff $\mathsf{mem}(N, \mathcal{S}(\sigma))$. Both sides are given coinductive characterizations:

$$\frac{\mathsf{mem}(M, \underline{\mathcal{S}}(\Gamma, x : A \Rightarrow B)) \quad \underline{\mathcal{S}}(\Gamma, x : A \Rightarrow B) \neq \mathbb{O}}{\mathsf{mem}(\lambda x.M, \underline{\mathcal{S}}(\Gamma \Rightarrow A \supset B))} \; (\underline{a})$$

$$\frac{\exists (y : \vec{B} \supset p) \underline{\in} \Gamma \; \forall i, \; \mathsf{mem}(M_i, \underline{\mathcal{S}}(\Gamma \Rightarrow B_i))}{\mathsf{mem}(y\langle M_i \rangle_i, \underline{\mathcal{S}}(\Gamma \Rightarrow p))} \; (\underline{b})$$

$$\frac{\mathsf{mem}(M, \mathcal{S}(\Gamma, x : A \Rightarrow B))}{\mathsf{mem}(\lambda x.M, \mathcal{S}(\Gamma \Rightarrow A \supset B))} \; (a)$$

$$\frac{\exists (y : \vec{B} \supset p) \in \Gamma \; \forall i, \; \mathsf{mem}(M_i, \mathcal{S}(\Gamma \Rightarrow B_i))}{\mathsf{mem}(y\langle M_i \rangle_i, \mathcal{S}(\Gamma \Rightarrow p))} \; (b)$$

These two coinductive definitions are extensionally equal, proven by coinduction in both directions (uses part 1).

Proof of part 3

Uses the two other parts and a general lemma about forests: if there is no solution, then there is an empty sum.

König's lemma for simple types

Theorem

For all sequents σ, the pruned solution space of σ is infinite iff σ has an infinite solution.

Needs all parts of the previous lemma and a general lemma about forests: if there is no empty sum, then finiteness ("by definition") is equivalent to all solutions being finite.

König's lemma for simple types

Theorem

For all sequents σ, the pruned solution space of σ is infinite iff σ has an infinite solution.

Needs all parts of the previous lemma and a general lemma about forests: if there is no empty sum, then finiteness ("by definition") is equivalent to all solutions being finite.

For the motivating example $x : p \supset q \supset p \Rightarrow p$, the pruned solution space is just the empty sum.

Final remarks

- new approach to inhabitation-like problems in STLC:
  - a single λ-term represents the entire search space of inhabitants of a type;
  - driven by the term syntax (hence by structure): description of inhabitation problems; a compact representation of solution spaces with an associated "König's lemma"

Final remarks

- new approach to inhabitation-like problems in STLC:
  - a single λ-term represents the entire search space of inhabitants of a type;
  - driven by the term syntax (hence by structure): description of inhabitation problems; a compact representation of solution spaces with an associated "König's lemma"
- the approach seems applicable to other logics enjoying the subformula property (e. g., with other propositional connectives);

Final remarks

- new approach to inhabitation-like problems in STLC:
  - a single λ-term represents the entire search space of inhabitants of a type;
  - driven by the term syntax (hence by structure): description of inhabitation problems; a compact representation of solution spaces with an associated "König's lemma"
- the approach seems applicable to other logics enjoying the subformula property (e. g., with other propositional connectives);
- inhabitation in STLC has been approached through various other means, including:
  - automata and languages (Takahashi, Hirokawa, Schubert, ...);
  - game semantics (Bourreau, Salvati);
  - formula-tree method (Broda, Damas, Alves);
  - sets of polynomial equations (Zaionc, David).
  - pre-grammar of a type (Alves, Broda at FSCD 2018).

THANK YOU!