Finiteness properties of simple types through a coinductive lambda-calculus

José Espírito Santo[1] Ralph Matthes[2] Luís Pinto[1]

[1] Centro de Matemática, Univ. Minho, Portugal

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

Third Workshop on Mathematical Logic and its Applications – MLA 2019

March 11, 2019
Loria, Vandoeuvre-lès-Nancy, France

Abstract

Proofs in propositional logic correspond to lambda-terms with simple types. The search for proofs of a given formula/type corresponds to the search for lambda-terms of that type. Finite successful runs of the search correspond to the construction of inhabitants of that type, but proof search is more than the set of its successful outcomes.
In previous work, we developed a representation of the search space for locally correct applications of the proof rules, not limiting the representation to the construction of (finite) inhabitants. The data structure we propose is an extension of lambda-calculus (restricted to long normal forms) to a coinductive structure that also allows to express choice points in the search process. The elements of that structure are called forests, and individual inductive or even coinductive lambda-terms can be seen as members of such a forest.

Abstract (2)

The forests form a coinductive datatype and are therefore not necessarily finitely described objects. The finitary counterpart to forests is 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.
The methodology suggested by these structures is to specify proof search problems through forests and by solving them effectively through finitary forests.
In particular, we used this to reprove decidability of inhabitation and type finiteness (i. e., the property of having only finitely many inhabitants), but also the prediction of at most one inhabitant or type finiteness based on the absence of two atom occurrences at negative resp. positive position in the formula.

Abstract (3)

We now come to as of now unreleased material. Type finiteness "naturally" means having only finitely many inhabitants. But, as soon as one grants the status of "member" of a formula A to any member of the forest canonically generated from search for proofs of A, other natural concepts of finiteness are possible. One is the finiteness of any member of A. 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.
The interesting technical observation here is that all three finiteness concepts are instances of a generic finiteness predicate with a set parameter, and their decidability is seen by a single proof for that predicate.

Abstract (4)

Time permitting, we will also discuss "pruning" of the search space generated from a formula, where all failed runs are chopped off. Our final result, which we dub König's lemma for simple types, says that finiteness of all members of a formula is equivalent to finiteness of the pruned search space generated from the formula.

Overview

- 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

- 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

- 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:

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

The decontraction phenomenon

(going by the name of cocontraction in our earlier papers)

- When searching for solutions in a given context Γ that contains two variables $x : A$ and $y : A$ with the same type A, then, whenever a coterm of a given type B can be constructed using assumption x, another coterm of that type B can be constructed with y in place of x, and this choice between x and y can be operated for each occurrence of x independently. (This phenomenon already occurs in the search for inhabitants, i. e., finite terms, but it challenges the understanding of regularity.)

The decontraction phenomenon

(going by the name of cocontraction in our earlier papers)

- When searching for solutions in a given context Γ that contains two variables $x : A$ and $y : A$ with the same type A, then, whenever a coterm of a given type B can be constructed using assumption x, another coterm of that type B can be constructed with y in place of x, and this choice between x and y can be operated for each occurrence of x independently. (This phenomenon already occurs in the search for inhabitants, i. e., finite terms, but it challenges the understanding of regularity.)
- The forests as a data structure are "unaware" of that phenomenon, but when forests are constructed to represent solution spaces of sequents, one has to be aware of these possibilities.

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:

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

The decontraction operation

We want to express that the canonical search space $\mathcal{S}(\sigma)$ fits the decontraction phenomenon.

Write $\Gamma \leq \Gamma'$ when Γ' is an inessential extension of context Γ (i. e., $\Gamma \subseteq \Gamma'$ but all formulas of Γ' are in Γ).

The decontraction operation

We want to express that the canonical search space $\mathcal{S}(\sigma)$ fits the decontraction phenomenon.

Write $\Gamma \leq \Gamma'$ when Γ' is an inessential extension of context Γ (i. e., $\Gamma \subseteq \Gamma'$ but all formulas of Γ' are in Γ).

For $\Gamma \leq \Gamma'$, the decontraction operation on forests $[\Gamma'/\Gamma]T$, which essentially adds in T more elimination alternatives due to more variables in Γ', is corecursively given by:

Definition (decontraction of contexts for expressions of λ_{Σ}^{co})

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

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

The canonical search space fits the decontraction phenomenon

Decontraction readily extends to sequents: for $\sigma = (\Gamma \Rightarrow A)$ and $\sigma' = (\Gamma' \Rightarrow A')$,

1. $\sigma \leq \sigma'$ if $\Gamma \leq \Gamma'$ and $A = A'$ (only for identical types);
2. if $\sigma \leq \sigma'$, then $[\sigma'/\sigma]T := [\Gamma'/\Gamma]T$

Lemma

If $\Gamma \leq \Gamma'$ *then* $\mathcal{S}(\Gamma' \Rightarrow A) = [\Gamma'/\Gamma](\mathcal{S}(\Gamma \Rightarrow A))$.
In other words:
If $\sigma \leq \sigma'$ *then* $\mathcal{S}(\sigma') = [\sigma'/\sigma](\mathcal{S}(\sigma))$.

Beware that the equation is meant to be bisimulation modulo taking sets of elimination alternatives.

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 $\sigma \leq \sigma'$, i. e., σ' may have more declarations than σ, but not with new types;

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

- 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 $\sigma \leq \sigma'$, i. e., 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 decontraction operation and the fixed-point construction operate on the meta-level (on forests).

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 $\lambda^{\mathrm{gfp}}_{\Sigma}$-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;

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;

  (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 $\lambda_{\Sigma}^{\mathrm{gfp}}$-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;

  (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.

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.

Decision and counting problems

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 $\mathrm{EF}_\emptyset(\mathcal{F}(\sigma))$.

Specifying the type finiteness problem

- Question: are there only finitely many inhabitants for a given sequent σ?
- Known to be decidable, even PSPACE-complete by a reduction to the inhabitation problem (Sachio Hirokawa).
- On the (semantic) level of forests, again expressed by studying the members of the forest.
- Finiteness of the number of finite members can be inductively characterized by a predicate finfin on forests, not shown here.
- The spec. through forests: finfin($\mathcal{S}(\sigma)$) iff σ is enjoys type finiteness.

Effectively solving the type finiteness problem

- syntax-directed inductive predicate FF_P defined as counterpart on finitary terms; in particular:

$$\frac{P(\sigma)}{\mathsf{FF}_P(X^\sigma)} \qquad \frac{\mathsf{FF}_P(N)}{\mathsf{FF}_P(\lambda x^A.N)} \qquad \frac{\forall i,\, \mathsf{FF}_P(E_i)}{\mathsf{FF}_P(\mathrm{gfp} X^\sigma.\sum_i E_i)}$$

$$\frac{\forall i,\, \mathsf{FF}_P(N_i)}{\mathsf{FF}_P(x\langle N_i\rangle_i)} \qquad \frac{\mathsf{NEF}_\star(N_j)}{\mathsf{FF}_P(x\langle N_i\rangle_i)}$$

  Crucially, $\mathsf{NEF}_\star$ is the canonical NEF, i. e., NEF_P with $P := \mathsf{EF}_\emptyset \circ \mathcal{F}$;

  Again, also its negation NFF_P can be given in such an inductive form. These predicates will later be generalized with a further parameter that abstracts from the specific situation of the last rule.

Effectively solving the type finiteness problem

- syntax-directed inductive predicate FF_P defined as counterpart on finitary terms; in particular:

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

$$\frac{\forall i,\ \mathrm{FF}_P(N_i)}{\mathrm{FF}_P(x\langle N_i\rangle_i)} \qquad \frac{\mathrm{NEF}_\star(N_j)}{\mathrm{FF}_P(x\langle N_i\rangle_i)}$$

  Crucially, $\mathrm{NEF}_\star$ is the canonical NEF, i. e., NEF_P with $P := \mathrm{EF}_\emptyset \circ \mathcal{F}$;

  Again, also its negation NFF_P can be given in such an inductive form. These predicates will later be generalized with a further parameter that abstracts from the specific situation of the last rule.

- the characterizations are equivalent:
  $\mathrm{FF}_P(T)$ iff $\mathrm{finfin}(\llbracket T \rrbracket)$, for $P \subseteq \mathrm{finfin} \circ \mathcal{S}$ and T of form $\mathcal{F}(\sigma)$;
- type finiteness for sequent σ decided by deciding $\mathrm{FF}_\emptyset(\mathcal{F}(\sigma))$.

Counting function

For a finitely inhabited type A (as decided on the previous slide), $\#(\mathcal{F}(\Rightarrow A))$ counts the number of inhabitants of A, where $\#$ is a very simple recursion over terms:

$$\begin{array}{rcl} \#(X^\sigma) & := & 0 \\ \#(\lambda x^A.N) & := & \#(N) \\ \#(\mathrm{gfp} X^\sigma.\sum_i E_i) & := & \sum_i \#(E_i) \\ \#(x\langle N_i\rangle_i) & := & \prod_i \#(N_i) \end{array}$$

Notice that we do not only add and multiply zeros since $\#(x\langle\rangle) = 1$.

Predicting the number of inhabitants

Predicting the number of inhabitants

What we mean by predicting the number: to give criteria on types that allow to infer a constraint on the number of inhabitants, such as those expressed by the predicates we have already seen: exfin and finfin and their negations, but also others:

Predicting the number of inhabitants

What we mean by predicting the number: to give criteria on types that allow to infer a constraint on the number of inhabitants, such as those expressed by the predicates we have already seen: exfin and finfin and their negations, but also others:

1. there is at most one inhabitant (coherence); its negation: there are at least two (different) inhabitants

Predicting the number of inhabitants

What we mean by predicting the number: to give criteria on types that allow to infer a constraint on the number of inhabitants, such as those expressed by the predicates we have already seen: exfin and finfin and their negations, but also others:

1. there is at most one inhabitant (coherence); its negation: there are at least two (different) inhabitants
2. if there is an inhabitant, then there are infinitely many inhabitants; its negation: the number of inhabitants is neither 0 nor infinite

Monatomic Thm. (Ben-Yelles 1979): Let $A = \vec{A} \supset p$ be a monatomic type. If all A_i's are p, the number of inhabitants of A is the length of $\vec{A}$; otherwise A has either 0 or infinitely many inhabitants.

This is a property of the second kind. It is easy to establish (also using our definitions).

Coherence theorem and its refinements

Coherence Thm. (Mints 1979): If no atom occurs positively in A more than once, and no atom occurs negatively in A more than once, then A has at most one inhabitant.
The first condition was shown to be redundant (Aoto and Ono 1994), so the result is:
Coherence Thm. (Mints 1979, Aoto and Ono 1994): If no atom occurs negatively in A more than once, then A has at most one inhabitant.
This was also proved by a game semantics approach by Bourreau and Salvati in 2011.

Coherence theorem and its refinements

Coherence Thm. (Mints 1979): If no atom occurs positively in A more than once, and no atom occurs negatively in A more than once, then A has at most one inhabitant.
The first condition was shown to be redundant (Aoto and Ono 1994), so the result is:
Coherence Thm. (Mints 1979, Aoto and Ono 1994): If no atom occurs negatively in A more than once, then A has at most one inhabitant.
This was also proved by a game semantics approach by Bourreau and Salvati in 2011.

What is the consequence of the omitted first condition alone?
Finiteness theorem for positively non-duplicated types (Broda and Damas 2005): If no atom occurs positively in A more than once, then A has only finitely many inhabitants (possibly none).

Example for prediction with the refinement of coherence

Let p, q, r be different atoms.
Consider $A := (p^+ \supset q^+ \supset r^-) \supset (p^+ \supset q^-) \supset p^- \supset r^+$.
We marked polarity of the atom occurrences. Each atom has only one negative occurrence. The refined coherence theorem applies, but not the original one by Mints since p has two positive occurrences. The prediction is thus that A has at most one inhabitant.

Example for prediction with the refinement of coherence

Let p, q, r be different atoms.
Consider $A := (p^+ \supset q^+ \supset r^-) \supset (p^+ \supset q^-) \supset p^- \supset r^+$.
We marked polarity of the atom occurrences. Each atom has only one negative occurrence. The refined coherence theorem applies, but not the original one by Mints since p has two positive occurrences. The prediction is thus that A has at most one inhabitant.

Of course, A is inhabitated by the S combinator.

Example for prediction with the refinement of coherence

Let p, q, r be different atoms.
Consider $A := (p^+ \supset q^+ \supset r^-) \supset (p^+ \supset q^-) \supset p^- \supset r^+$.
We marked polarity of the atom occurrences. Each atom has only one negative occurrence. The refined coherence theorem applies, but not the original one by Mints since p has two positive occurrences. The prediction is thus that A has at most one inhabitant.

Of course, A is inhabitated by the S combinator.

Already in the monatomic case, there are easy examples that show that two occurrences of the same polarity are problematic:

- $p^- \supset p^- \supset p^+$ is not coherent, with two negative occurrences
- $(p^+ \supset p^-) \supset (p^- \supset p^+)$ has infinitely many inhabitants, the Church numerals, with two positive occurrences (but also needs two negative occurrences since otherwise the coherence thm. would apply)

Our method allows to reprove these results.

We claim to have perspicuous new proofs of those (old) results, and they are obtained by exploring the structure of the finitary representation of solution spaces.

Our method allows to reprove these results.

We claim to have perspicuous new proofs of those (old) results, and they are obtained by exploring the structure of the finitary representation of solution spaces.

We identify simple properties of finitary forests:

- a finitary forest may have no occurrence of fixed-point variables (which intuitively guarantees finiteness)
- a finitary forest may have no choice points with at least two options (which intuitively guarantees uniqueness)

Our method allows to reprove these results.

We claim to have perspicuous new proofs of those (old) results, and they are obtained by exploring the structure of the finitary representation of solution spaces.

We identify simple properties of finitary forests:

- a finitary forest may have no occurrence of fixed-point variables (which intuitively guarantees finiteness)
- a finitary forest may have no choice points with at least two options (which intuitively guarantees uniqueness)

We prove that these properties hold for $\mathcal{F}(\Rightarrow A)$ if A is positively non-duplicated resp. negatively non-duplicated (under an extra proviso).

Our method allows to reprove these results.

We claim to have perspicuous new proofs of those (old) results, and they are obtained by exploring the structure of the finitary representation of solution spaces.

We identify simple properties of finitary forests:

- a finitary forest may have no occurrence of fixed-point variables (which intuitively guarantees finiteness)
- a finitary forest may have no choice points with at least two options (which intuitively guarantees uniqueness)

We prove that these properties hold for $\mathcal{F}(\Rightarrow A)$ if A is positively non-duplicated resp. negatively non-duplicated (under an extra proviso).

The easy part is to infer finiteness resp. uniqueness, by using our predicates on finitary forests to decide inhabitation.

Reproving the Broda-Damas theorem on finiteness

Three steps:

1. Call T strongly acyclic if T has no occurrence, free or bound, of fixed-point variables.
2. Show by term induction that if T is strongly acyclic then $\mathrm{FF}_{\emptyset}(T)$.
3. Prove that if no atom occurs positively more than once in A, then $\mathcal{F}(\Rightarrow A)$ is strongly acyclic.

Reproving the Broda-Damas theorem on finiteness

Three steps:

1. Call T strongly acyclic if T has no occurrence, free or bound, of fixed-point variables.
2. Show by term induction that if T is strongly acyclic then $\mathrm{FF}_{\emptyset}(T)$.
3. Prove that if no atom occurs positively more than once in A, then $\mathcal{F}(\Rightarrow A)$ is strongly acyclic.

The last step is difficult since $\mathcal{F}(\Rightarrow A) = \mathcal{F}(\Rightarrow A; \emptyset)$ is defined by recursion involving non-empty vectors of declarations in place of $\emptyset$. A proof by induction on $\mathcal{F}(\sigma; \Xi)$ will show its strong acyclicity given that a certain invariant P holds of σ and Ξ. P is designed to be equivalent to the condition of positive non-duplication in the case of $\Rightarrow A$ and $\emptyset$.

Reproving the coherence theorem

Same methodology as on the previous slide:

1. Call T deterministic if every sum in T has at most one summand.
2. Show by term induction that if T is deterministic, then $\#(T) \leq 1$ and, in addition, $\mathrm{EF}_{\emptyset}(T)$ implies $\mathrm{FF}_{\emptyset}(T)$.
3. Prove that if no atom occurs negatively more than once in A and $\mathrm{EF}_{\emptyset}(\mathcal{F}(\Rightarrow A))$, then $\mathcal{F}(\Rightarrow A)$ is deterministic.

Reproving the coherence theorem

Same methodology as on the previous slide:

1. Call T deterministic if every sum in T has at most one summand.
2. Show by term induction that if T is deterministic, then $\#(T) \leq 1$ and, in addition, $\mathsf{EF}_\emptyset(T)$ implies $\mathsf{FF}_\emptyset(T)$.
3. Prove that if no atom occurs negatively more than once in A and $\mathsf{EF}_\emptyset(\mathcal{F}(\Rightarrow A))$, then $\mathcal{F}(\Rightarrow A)$ is deterministic.

The proof in the last step is more difficult since intermediate results are needed, and not just a generalization to general $\mathcal{F}(\sigma; \Xi)$. Still, all proofs work by recursion on the build-up of $\mathcal{F}(\sigma; \Xi)$. Two intermediate results address strong acyclicity in certain situations.

Again, the theorem easily follows from the last step.

Reproving the coherence theorem

Same methodology as on the previous slide:

1. Call T deterministic if every sum in T has at most one summand.
2. Show by term induction that if T is deterministic, then $\#(T) \leq 1$ and, in addition, $\mathrm{EF}_{\emptyset}(T)$ implies $\mathrm{FF}_{\emptyset}(T)$.
3. Prove that if no atom occurs negatively more than once in A and $\mathrm{EF}_{\emptyset}(\mathcal{F}(\Rightarrow A))$, then $\mathcal{F}(\Rightarrow A)$ is deterministic.

The proof in the last step is more difficult since intermediate results are needed, and not just a generalization to general $\mathcal{F}(\sigma; \Xi)$. Still, all proofs work by recursion on the build-up of $\mathcal{F}(\sigma; \Xi)$. Two intermediate results address strong acyclicity in certain situations.

Again, the theorem easily follows from the last step.

Consider $A := ((((r^- \supset p^+) \supset q^-) \supset q^+) \supset p^-) \supset p^+$. $\mathcal{F}(\Rightarrow A)$ is not deterministic although the condition on occurrences is met.

The new material

The concepts and results presented so far are accepted for publication in MSCS, already available online at CUP since April 2018, see `https://doi.org/10.1017/S0960129518000099`.

The new material

The concepts and results presented so far are accepted for publication in MSCS, already available online at CUP since April 2018, see `https://doi.org/10.1017/S0960129518000099`.

- 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).

The new material

The concepts and results presented so far are accepted for publication in MSCS, already available online at CUP since April 2018, see `https://doi.org/10.1017/S0960129518000099`.

- 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.

The new material

The concepts and results presented so far are accepted for publication in MSCS, already available online at CUP since April 2018, see `https://doi.org/10.1017/S0960129518000099`.

- 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

- Inductively defined by

$$\frac{\neg\Pi(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\mathrm{fin}^{\Pi}(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(E_i)}{\mathrm{fin}^{\Pi}(\sum_i E_i)}$$

$$\frac{\neg\Pi(N_j)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(N_i)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)}$$

The generic notion of finiteness for forests

- Inductively defined by

$$\frac{\neg\Pi(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\mathrm{fin}^{\Pi}(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(E_i)}{\mathrm{fin}^{\Pi}(\sum_i E_i)}$$

$$\frac{\neg\Pi(N_j)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(N_i)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)}$$

- finfin (used for type finiteness) is $\mathrm{fin}^{\mathrm{exfin}}$.

The generic notion of finiteness for forests

- Inductively defined by

$$\frac{\neg\Pi(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\mathrm{fin}^{\Pi}(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(E_i)}{\mathrm{fin}^{\Pi}(\sum_i E_i)}$$

$$\frac{\neg\Pi(N_j)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(N_i)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)}$$

- finfin (used for type finiteness) is $\mathrm{fin}^{\mathrm{exfin}}$.
- Set allfin $:= \mathrm{fin}^{\mathrm{exsol}}$, and this represents finiteness in the sense of having only finite solutions, as can be shown by coinduction about the complements.

The generic notion of finiteness for forests

- Inductively defined by

$$\frac{\neg\Pi(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\mathrm{fin}^{\Pi}(N)}{\mathrm{fin}^{\Pi}(\lambda x^{A}.N)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(E_i)}{\mathrm{fin}^{\Pi}(\sum_i E_i)}$$

$$\frac{\neg\Pi(N_j)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)} \qquad \frac{\forall i,\ \mathrm{fin}^{\Pi}(N_i)}{\mathrm{fin}^{\Pi}(x\langle N_i\rangle_i)}$$

- finfin (used for type finiteness) is $\mathrm{fin}^{\mathrm{exfin}}$.
- Set allfin $:= \mathrm{fin}^{\mathrm{exsol}}$, and this represents finiteness in the sense of having only finite solutions, as can be shown by coinduction about the complements.
- If the parameter Π contains all forests, fin^{Π} captures the forests that are inductively generated by its grammar, they are "finite by definition" – the smallest notion we get as instance.

The generic notion of finiteness for finitary forests

- Inductive definition

$$\frac{P(\sigma)}{\mathsf{F}^{\Pi}_{P}(X^{\sigma})} \qquad \frac{\mathsf{F}^{\Pi}_{P}(N)}{\mathsf{F}^{\Pi}_{P}(\lambda x^{A}.N)} \qquad \frac{\forall i,\ \mathsf{F}^{\Pi}_{P}(E_i)}{\mathsf{F}^{\Pi}_{P}(\mathrm{gfp} X^{\sigma}.\sum_i E_i)}$$

$$\frac{\forall i,\ \mathsf{F}^{\Pi}_{P}(N_i)}{\mathsf{F}^{\Pi}_{P}(x\langle N_i\rangle_i)} \qquad \frac{\neg\Pi([\![N_j]\!]^{s})}{\mathsf{F}^{\Pi}_{P}(x\langle N_i\rangle_i)}$$

for P satisfying the proviso: $P \subseteq \mathrm{fin}^{\Pi} \circ \mathcal{S}$ and P decidable.

The generic notion of finiteness for finitary forests

- Inductive definition

$$\frac{P(\sigma)}{\mathrm{F}^{\Pi}_{P}(X^{\sigma})} \qquad \frac{\mathrm{F}^{\Pi}_{P}(N)}{\mathrm{F}^{\Pi}_{P}(\lambda x^{A}.N)} \qquad \frac{\forall i,\ \mathrm{F}^{\Pi}_{P}(E_i)}{\mathrm{F}^{\Pi}_{P}(\mathrm{gfp} X^{\sigma}.\sum_i E_i)}$$

$$\frac{\forall i,\ \mathrm{F}^{\Pi}_{P}(N_i)}{\mathrm{F}^{\Pi}_{P}(x\langle N_i\rangle_i)} \qquad \frac{\neg\Pi([\![N_j]\!]^{s})}{\mathrm{F}^{\Pi}_{P}(x\langle N_i\rangle_i)}$$

  for P satisfying the proviso: $P \subseteq \mathrm{fin}^{\Pi} \circ \mathcal{S}$ and P decidable.
- We need to use the "simplified semantics" $[\![T]\!]^{s}$ for some subterms T, which was hidden in the decidability proofs, but here even appears in the definition of the concept. In our leading examples, decidability results are available to make this definition effective.
- Our results for type finiteness can be generalized to sets Π that are closed under decontraction both ways, for which $\neg\Pi \subseteq \mathrm{fin}^{\Pi}$ and where the applicability of the last clause of the above definition is decidable.

Strengthening the Broda-Damas result

We can show that strongly acyclic finitary forests are precisely those for which $\mathsf{F}_{\emptyset}^{\Pi}$ holds with Π all the forests.

Strengthening the Broda-Damas result

We can show that strongly acyclic finitary forests are precisely those for which $\mathsf{F}_{\emptyset}^{\Pi}$ holds with Π all the forests.

The original result was: If no atom occurs positively in A more than once, then A has only finitely many inhabitants (possibly none).

We now get: If no atom occurs positively in A more than once, then the search space for A is finite, which in particular implies that A has only finitely many inhabitants and only finite solutions.

Pruning the solution space

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

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, $\mathrm{ES}_\star(\mathcal{F}(\Delta \Rightarrow B_j))$.

Then, the pruned solution space has the same members as the original one; if σ has a solution, the pruned space has no empty sum, otherwise, it is the empty sum.

König's lemma for simple types

We can then easily show: For all sequents σ, the pruned solution space of σ is infinite iff σ has an infinite solution.

König's lemma for simple types

We can then easily show: For all sequents σ, the pruned solution space of σ is infinite iff σ has an infinite solution.

This would not be true for the original definition of solution space: $x : p \supset q \supset p \Rightarrow p$ has no solution but an infinite solution space. Of course, 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; recursive functions for deciding the problems and counting inhabitants; new proofs of two theorems on the prediction of the number of inhabitants based on the absence of multiple occurrences of atoms of the same polarity;

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; recursive functions for deciding the problems and counting inhabitants; new proofs of two theorems on the prediction of the number of inhabitants based on the absence of multiple occurrences of atoms of the same polarity;
- 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; recursive functions for deciding the problems and counting inhabitants; new proofs of two theorems on the prediction of the number of inhabitants based on the absence of multiple occurrences of atoms of the same polarity;
- 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!