Deciding unique inhabitants with sums (work in progress)
Gabriel Scherer, Didier Rémy

To cite this version:
Gabriel Scherer, Didier Rémy. Deciding unique inhabitants with sums (work in progress). TYPES, May 2014, Paris, France. hal-01094127

HAL Id: hal-01094127
https://inria.hal.science/hal-01094127
Submitted on 18 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Deciding unique inhabitants with sums
(work in progress)

Gabriel Scherer, Didier Rémy

INRIA – Gallium

Our ongoing work focuses on types that have a unique inhabitant—modulo program equivalence. If we were able to detect when such types appear in a program, we could perform program inference to relieve the programmer from the obligation to write the less-interesting parts of the program. Unicity of inhabitant is a strong form of principality: inference cannot make a “wrong” guess as there is only one choice. This property can be contrasted to two different notions, inhabited types and subtyping coercions, that it respectively refines and extends:

- The usual provability notion of “having at least an inhabitant”, which is the one of concern when using strongly-typed lambda-calculi to prove mathematical facts. In this setting, the dynamic semantics of terms is ignored, which makes it unsuited to program inference.
- The common programming notion of erasable subtyping between two types \( A \leq B \): inference systems for subtyping can be seen as a restrictive type system for functions whose terms have a computational interpretation which is always the identity function—when it is inhabited, \( A \leq B \) has a unique inhabitant. On the contrary, some types have unique inhabitants that are not the identity functions, such as \( \text{swap} : \forall AB. (A*B) \rightarrow (B*A) \).

To decide uniqueness, we must be able to enumerate the distinct terms at a given type.

As a first step, we consider the simply-typed lambda-calculus with arrows, product and sums. We are looking for a term enumeration process that is complete, i.e., it does not miss any computational behavior, and canonical, i.e., it has no duplicates. We propose an approach based on saturation, with encouraging results, although termination is still a conjecture.

**Computational completeness** Some existing proof calculi, such as *contraction-free calculi* [Dyc13], make simplifications designed to make provability decision tractable, but throw away some behaviors, loosing computational completeness. The following rule, which drops a function after its first invocation, preserves provability even in a contraction-free calculus:

\[
\frac{\Gamma, A \rightarrow B \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \rightarrow B \vdash C}
\]

Consider a datatype \( A \ast (A \rightarrow (A \ast B)) \) of infinite streams of \( B \) with internal state \( A \). If the generating function is dropped after its first result, there is exactly one proof of \( A \ast (A \rightarrow (A \ast B)) \vdash B \) (getting the first element of the stream), while there are infinitely many observably distinct programs at that type.

**Focusing** Focusing [And92] imposes a phase discipline on derivations by distinguishing invertible and non-invertible inference rules—invertible rules are those whose inverse is derivable. In absence of sums, focused proofs are in exact correspondence to \( \beta \)-short \( \eta \)-long proof terms; it is computationally complete. Starting from a grammar for values and neutrals:

\[
v ::= \lambda x : A. v \quad | \quad (v, v) \quad | \quad n
\]

\[
n ::= n v \quad | \quad \pi_1 n \quad | \quad \pi_2 n \quad | \quad x
\]

we can define the set \( \text{Val}(\Gamma \vdash A) \) of (distinct) values of type \( A \) in the environment \( \Gamma \), along with the set \( \text{Ne}(\Gamma \vdash A) \) of neutrals at this type, by lifting notations from terms to sets of terms:
Val(Γ ⊢ A → B) := λ(x:A)Val(Γ, x:A ⊢ B)
Val(Γ ⊢ A + B) := (Val(Γ ⊢ A), Val(Γ ⊢ B))
Val(Γ ⊢ X) := Ne(Γ ⊢ X) (X atomic) Ne(Γ ⊢ N) ⊇ \{x | (x:N) ∈ Γ\}
Ne(Γ ⊢ A) ⊇ π1 Ne(Γ ⊢ A1 + A2)
Ne(Γ ⊢ A → B) Val(Γ ⊢ A)

Note that Val is structurally recursive on the input type; it corresponds to the invertible rules.

Lemma 1 (Canonicity of negative neutrals). For any n1, n2 ∈ Ne(Γ ⊢ A), if n1 and n2 are syntactically distinct, then they are distinct for contextual equivalence.

This specification can be turned into a decision procedure; termination arguments are of two sorts. First, the subformula property gives a finite bound on the number of types that will be considered. Second, cycles in the equations defining Ne (in particular types with an infinite number of distinct inhabitants, such as Church integers X → (X → X) → X) can be worked upon using a graph-based representation in the style of Wells and Yakobowski [WY04].

Unfortunately, focusing alone does not capture the notably difficult η-equivalence for sums. Writing δ(e1, x, e2, y, e3) for (match e1 with inl x → e2 | inr y → e3), the two following terms correspond to distinct, but observationally equivalent, focused proofs of (1 → A + B) → A + B: (λ(f) δ(f 1, x, inl x, y, inr y)) and (λ(f) δ(f 1, x, inl x, y.δ(f 1, x′, in1 x′, y′, ι1 y′))).

Saturation Coupling an enumeration of focused proofs with an equivalence checking procedure for sums [Lin07] does not work, as there may be infinitely many redundant copies. The main idea of these algorithms is to move sum elimination as high in the term as possible. We thus propose to eliminate sums as early as possible during term generation, by simultaneously eliminating all neutrals of any positive type P when the goal is itself a positive Q:

Val(Γ, x:(A + B) ⊢ C) := δ(x, y, Val(Γ, y:A ⊢ C), z, Val(Γ, z:B ⊢ C))
Ne(Γ ⊢ A1 + A2) ⊇ ∪i∈\{1,2\} σi Ne(Γ ⊢ Ai)
Val(Γ ⊢ Q) := Ne(Γ ⊢ Q) ∪ (let Δ = (∪P Ne(Γ ⊢ P)) in Val(Γ, Δ ⊢ Q))

This saturation process is complete and canonical. Remarkably, it is strongly related to the notion of “maximal multi-focusing” [CMS08]: our proofs are also maximal in this sense.

While completeness and non-duplicability are relatively simple, termination is still a conjecture. In addition to the “inner” non-termination related to the definition of Ne(Γ ⊢ A) (infinite number of distinct terms), there is now an “outer” termination problem of infinite alternance of Val and Ne layers. Previous approaches gave either completeness and termination (focusing, with duplicates) or canonicity and termination (incomplete provability calculi), while this presentation is naturally complete and canonical, but not yet proved terminating.

References