Type-Based Productivity of Stream Definitions in the Calculus of Constructions

Jorge Luis Sacchini
Carnegie Mellon University — Doha, Qatar
Email: sacchini@qatar.cmu.edu

***Abstract*—Productivity of corecursive definitions is an essential property in proof assistants since it ensures logical consistency and decidability of type checking. Type-based mechanisms for ensuring productivity use types annotated with size information to track the number of elements produced in corecursive definitions. In this paper, we propose an extension of the Calculus of Constructions—the theory underlying the Coq proof assistant—with a type-based criterion for ensuring productivity of stream definitions. We prove strong normalization and logical consistency. Furthermore, we define an algorithm for inferring size annotations in types. These results can be easily extended to handle general coinductive types.**

I. Introduction

Coinductive types play an important role in proof assistants in modeling and reasoning about infinite data (such as streams) and infinite processes (such as communication protocols).

Coinductive types can be seen as the dual of inductive types. While recursive functions allow to analyze (consume) elements of inductive types, corecursive functions allow to *produce* elements of coinductive types.

Proof assistants based on dependent type theory impose restrictions for defining recursive and corecursive functions, in order to maintain logical consistency and decidability of type checking. For recursive functions, the requirement is termination. Dually, for corecursive functions, the requirement is productivity. In the case of stream definitions, productivity means that computing any element of the stream takes a finite amount of time. In other words, computation cannot get stuck.

Proof assistants, such as Coq [1] and Agda [2], use syntactic methods, so-called *guard predicates*, to ensure productivity and termination. Basically, a corecursive definition is accepted as productive if all recursive calls are *guarded* by a constructor. Let us illustrate with an example. Consider the function $\mathsf{nats}\ n$ that computes the stream $[n, n+1, \ldots]$. In Coq, we can define it by a cofixpoint definition as follows:

$$\mathsf{nats} \stackrel{\text{def}}{=} \mathsf{cofix\ nats} : \mathsf{nat} \to \mathsf{stream\,nat} := \lambda n.\ \mathsf{cons}(n, \mathsf{nats}\,(1+n))$$

where $\mathsf{cons} : A \to \mathsf{stream}\,A \to \mathsf{stream}\,A$ is the stream constructor. nats satisfies the guard predicate since the only corecursive call is *guarded* by cons. This ensures productivity.

However, this criterion is very restrictive. For example, the following alternative definition of nats is productive, but it is not accepted by Coq:

$$\mathsf{nats}' \stackrel{\text{def}}{=} \lambda n.\ \mathsf{cofix\ nats}' : \mathsf{stream\,nat} := \mathsf{cons}(n, \mathsf{map}\,(\lambda x.1+x)\,\mathsf{nats}')$$

where $\mathsf{map}\ F\ x$ is the stream obtained by applying F to each element of stream x. The reason is that, since nats' is the argument to a function, it is not considered guarded by cons.

The limitations of guard predicates appear often in practice, both for ensuring termination and productivity. Several alternatives have been proposed in the context of proof assistants. Type-based approaches are emerging as the ideal candidate to replace guard predicates in proof assistants based on dependent type theory, such as Coq and Agda. Type-based approaches offer several advantages over syntactic methods [3]–[6]. In particular, they are more expressive and easier to understand.

In previous work, Grégoire and the author defined an extension of the Calculus of Inductive Constructions (CIC) with a type-based termination mechanism similar to the system CIC$\hat{}$ [6] of Barthe et al. Our contribution was the completion of the metatheoretical study of type-based termination in CIC [5], [7]. Most important, we proved strong normalization and logical consistency.

In this work, we extend our previous results to the coinductive case. The contributions of this paper are the following.

- We define an extension of the Calculus of Constructions with universes, CCω, with a stream type and a type-based criterion for ensuring productivity of stream definitions (Sect. III).
- We prove strong normalization and logical consistency, using a model based on Λ-sets [8] (Sect. IV), fixing some errors in the sketch we proposed in [5].
- We give an algorithm for inferring sized types, based on the algorithm for CIC$\hat{}$ [6] (Sect. V).

II. Type-Based Productivity

In this section, we give a (very) short introduction to type-based mechanisms for ensuring productivity and termination. While several systems with different strengths have been proposed, they all share some common principles.

The main idea is to annotate types (inductive and coinductive) with size information. For inductive types, the size annotation represents an upper bound on the size of its elements. For example, the type $\mathsf{list}^s\,T$ represents the type of lists, of type T, whose length is *at most* s. On recursive functions, termination is ensured by allowing recursive calls only on smaller arguments—based on their type.

Dually, for coinductive types, the size annotation represent a *lower bound* on the number of elements produced. For example, the type $\mathsf{stream}^s\,T$ represents the type of streams

(infinite lists) of type T of which *at least* the first s elements can be produced. In our case, a size is either a variable, e.g. $\imath, \jmath$, the successor of a size, $\widehat{s}$, or ∞ which means there is no bound. Let us illustrate how productivity is ensured by looking at a simplified version of the typing rule for corecursion:

$$\frac{\Gamma(f : \mathsf{stream}^{\imath}\, T) \vdash M : \mathsf{stream}^{\widehat{\imath}}\, T}{\Gamma \vdash (\mathsf{cofix}\ f{:}\mathsf{stream}\, T := M) : \mathsf{stream}^{\infty}\, T}$$

This typing rule is read as follows: given a stream f of at least $\imath$ elements (i.e. the first $\imath$ elements can be produced), $M(f)$ returns a stream of $\imath + 1$ elements. By iterating M, we can produce all elements of the stream.

The advantage of this approach over guard predicates is that we can encode more information about a function in its type. For example, the map function on streams can be given the type $\mathsf{stream}^s\, A \to \mathsf{stream}^s\, B$. In turn, this allows to accept both definitions of nats given in the Introduction (see Sect. III-A for more examples).

III. CCω WITH STREAMS

In this section we introduce our system, called CC$\widehat{\omega}$, an extension of CCω with sized streams. CC$\widehat{\omega}$ is the extension with coinduction of the system we presented in [5], [7], which is itself closely related to CIC$\widehat{\ }$ [6].

The main feature of CC$\widehat{\omega}$ is, of course, the size annotations in streams (and coinductive types in general). Size (or stage) expressions are given by the following grammar:

$$\mathcal{S} ::= \mathcal{V}_{\mathcal{S}} \mid \widehat{\mathcal{S}} \mid \infty$$

where $\mathcal{V}_{\mathcal{S}}$ is a denumerable set of stage variables. We use $\imath, \jmath$ to denote stage variables, and s, r to denote stages. We write $\lfloor s \rfloor$, called the base of s, for the partial function defined by $\lfloor \imath \rfloor = \imath$ and $\lfloor \widehat{s} \rfloor = \lfloor s \rfloor$ (it is not defined for ∞).

We define a substage relation, denoted with $\sqsubseteq$, that we use for defining a subtyping relation. It is given by the reflexive-transitive closure of the relation defined by the rules $s \sqsubseteq \widehat{s}$ and $s \sqsubseteq \infty$, for all stages s.

We present CC$\widehat{\omega}$ in the style of Pure Type Systems [9] (PTS) by giving the specification of its sorts (universes). The set of sorts, denoted $\mathcal{U}$, includes an impredicative universe, Prop, and a predicative hierarchy $\{\mathsf{Type}_i\}_{i\geq 0}$. The specification of CC$\widehat{\omega}$ universes is completed by the sets Axiom (that describe the typing rules for sorts) and Rule (that describe the typing rules for products), defined as follows:

$$\begin{aligned}\mathsf{Axiom} &= \{(\mathsf{Prop}, \mathsf{Type}_0)\} \cup \{(\mathsf{Type}_i, \mathsf{Type}_{i+1})\}_{i\geq 0}\\ \mathsf{Rule} &= \{(\mathsf{Type}_i, \mathsf{Prop}, \mathsf{Prop})\}_{i\geq 0} \cup\\ &\qquad \{(\mathsf{Type}_i, \mathsf{Type}_j, \mathsf{Type}_{\mathsf{max}(i,j)})\}_{i,j\geq 0}\end{aligned}$$

As a PTS, CC$\widehat{\omega}$ is functional, meaning that Axiom and Rule can be seen as functions; concretely, $\mathsf{Axiom} : \mathcal{U} \to \mathcal{U}$ and $\mathsf{Rule} : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$—we exploit this view in the size-inference algorithm of Sect. V.

Following [6], [7], [10], we consider three classes of terms, which differ in the kind of annotations that (co-)inductive types carry. (The distinction is necessary to ensure subject reduction and efficient size inference.) In *bare terms*, (co-)inductive types carry no annotations. In *positions terms*, (co-)inductive types either carry no annotations, or a $\star$, used to indicate (co-)recursive positions in (co-)fixpoint definitions. Finally, in *sized terms*, (co-)inductive types carry a size expression.

Definition 1 (Terms)**.** *The terms of CC$\widehat{\omega}$ are given by the following generic grammar defined over a set a.*

$$\begin{aligned}\mathcal{T}[a] ::= {} & \mathcal{U} \mid \Pi x{:}\mathcal{T}[a].\mathcal{T}[a] \mid \mathcal{V} \mid \lambda x{:}\mathcal{T}^{\circ}.\mathcal{T}[a] \mid \mathcal{T}[a]\,\mathcal{T}[a]\\ & \mid \mathsf{stream}^{a}\,\mathcal{T}[a] \mid \mathsf{cons}\,(\mathcal{T}[a], \mathcal{T}[a])\\ & \mid \mathsf{case}_{\mathcal{T}^{\circ}}\ \mathcal{V} := \mathcal{T}[a]\ \mathsf{of}\ \mathsf{cons}(\mathcal{V}, \mathcal{V}) \Rightarrow \mathcal{T}[a]\\ & \mid \mathsf{cofix}\ \mathcal{V}{:}\mathcal{T}^{\star} := \mathcal{T}[a]\end{aligned}$$

where $\mathcal{V}$ is a denumerable set of term variables. The set of bare terms, position terms and sized terms are defined by $\mathcal{T}^{\circ} ::= \mathcal{T}[\epsilon]$, $\mathcal{T}^{\star} ::= \mathcal{T}[\{\epsilon, \star\}]$, and $\mathcal{T} ::= \mathcal{T}[\mathcal{S}]$, respectively. We also consider the class of sized terms with no size variables: $\mathcal{T}^{\infty} ::= \mathcal{T}[\infty]$.

We make use of the following erasure functions: $|\cdot| : \mathcal{T} \to \mathcal{T}^{\circ}$ (that erases all size annotations), $|\cdot|^{\imath} : \mathcal{T} \to \mathcal{T}^{\star}$ (that replaces with $\star$ all size annotations s such that $\lfloor s \rfloor = \imath$ and erases the rest), and $(\cdot)^{\infty} : \mathcal{T} \to \mathcal{T}^{\infty}$ (that replaces all size annotations with ∞). Given a term M, we write $\mathrm{FV}(M)$ for the set of free term variables of M and $\mathrm{SV}(M)$ for the set of size variables occurring in M. We omit the obvious definitions of these operators.

Let us briefly explain the constructions related to streams (the rest are standard). The constructor form $\mathsf{cons}(M, N)$ represents the stream whose head (first element) is M, and whose tail (the rest of the stream) is N. In Coq, with streams defined as a coinductive type, cons would have three arguments: the type of elements of the stream, the head, and the tail. Here, we can leave the type implicit and infer it from the head.

We can perform case analysis on a stream: in $\mathsf{case}_{P^{\circ}}\ x := M\ \mathsf{of}\ \mathsf{cons}(y_1, y_2)\ \Rightarrow\ N$, the argument, M, is an stream and P° is the return type (which can depend on x). Streams can be constructed by a cofixpoint construction of the form $\mathsf{cofix}\ f{:}T^{\star} := M$, where f can occur in M. The position type marks recursive positions (as illustrated in the examples below). Position types were introduced in [10] to ensure most general types. The typing rules ensure that $T^{\star}$ is a function type that returns (produces) a stream.

A *context* is a finite sequence of variable declarations, written $(x_1{:}T_1) \ldots (x_n{:}T_n)$. We use Γ, Δ to denote contexts. A bare—resp. position—context is a finite sequence of variable declarations of the form $(x{:}T^{\circ})$—resp. $(x{:}T^{\star})$.

Reduction: It is defined by β-reduction, $\imath$-reduction (pattern matching), and ν-reduction (cofixpoint unfolding). The first two are defined by the following rules:

$$(\lambda x{:}T^{\circ}.M)\, N \ \ \beta \ \ N\,[M/x]$$

$$\left(\begin{array}{l}\mathsf{case}_{P^{\circ}}\ x := \mathsf{cons}(M_1, M_2)\ \mathsf{of}\\ \mathsf{cons}(y_1, y_2) \Rightarrow N\end{array}\right) \ \ \imath \ \ N\,[M_1/y_1]\,[M_2/y_2]$$

In the case of cofixpoints, reduction needs to be restricted, as unrestricted cofixpoint unfolding is obviously not strongly

$$\frac{T\ R\ U}{T \preceq_R U} \qquad \frac{U_1 \preceq_R T_1 \qquad T_2 \preceq_R U_2}{\Pi x{:}T_1.T_2 \preceq_R \Pi x{:}U_1.U_2}$$

$$\frac{s \sqsubseteq r \qquad T \preceq_R U}{\mathsf{stream}^r\, T \preceq_R \mathsf{stream}^s\, U} \qquad \frac{T_1 \preceq_R T_2 \qquad T_2 \preceq_R T_3}{T_1 \preceq_R T_3}$$

Fig. 1. Subtyping relation

$$\frac{\imath \notin \mathrm{SV}(T)}{\imath\ \mathsf{occ}^{\xi}\ T} \qquad \frac{\imath\ \mathsf{occ}^{-\xi}\ T \qquad \imath\ \mathsf{occ}^{\xi}\ U}{\imath\ \mathsf{occ}^{\xi}\ \Pi x{:}T.U}$$

$$\frac{\imath \notin s \qquad \imath\ \mathsf{pos}\ T}{\imath\ \mathsf{pos}\ (\mathsf{stream}^s\, T)} \qquad \frac{\imath\ \mathsf{neg}\ T}{\imath\ \mathsf{neg}\ (\mathsf{stream}^s\, T)}$$

Fig. 2. Positivity of stage variables

normalizing. We use a lazy strategy where cofixpoints are only unfolded in the context of a case analysis. (At the end of this section we discuss how this choice affects the metatheory of CC$\widehat{\omega}$.) We define ν-reduction as follows:

$$\left(\begin{array}{l}\mathsf{case}_{P^\circ}\ x := (\mathsf{cofix}\ f{:}T^\star := M)\,\vec{N}\ \mathsf{of}\\ \mathsf{cons}(y_1, y_2) \Rightarrow O\end{array}\right)\ \nu$$
$$\left(\begin{array}{l}\mathsf{case}_{P^\circ}\ x := M\,[(\mathsf{cofix}\ f{:}T^\star := M)/f]\ \vec{N}\ \mathsf{of}\\ \mathsf{cons}(y_1, y_2) \Rightarrow O\end{array}\right)$$

where $\vec{N}$ is a sequence of terms. We write $\rightarrow$ for the compatible closure of $\beta\imath\nu$-reduction.

Typing: It is defined by the following judgments:

- $\mathsf{WF}(\Gamma)$ means that context Γ is well formed;
- $\Gamma \vdash M : T$ means that M has type T under context Γ.

The typing rules depend on a notion of subtyping and positivity of stage variables, defined by the following judgments:

- $T \preceq_R U$ means that T is a subtype of U with respect to a reduction relation R on terms;
- $\imath\ \mathsf{pos}\ T$ (resp. $\imath\ \mathsf{neg}\ T$) means that $\imath$ occurs positively (resp. negatively) in T.

Subtyping is defined by the rules of Fig. 1. The instances of R that we will use are convertibility ($\approx$) and α-equivalence ($\equiv$). We write $\leq$ for $\preceq_{\approx}$. Positivity of a stage variable is defined by the rules of Fig. 2. We use polarities to simplify the definition of positivity: a polarity ξ is either $+$ or $-$. Given a polarity ξ, we write $-\xi$ for the opposite polarity. We write occ^+ (resp. occ^-) to mean pos (resp. neg).

Subtyping in products is contravariant in the domain and covariant in the codomain. Size annotations occur negatively in coinductive types. E.g. $\mathsf{stream}^{\widehat{s}}\, T \leq \mathsf{stream}^s\, T$; in words: if a stream is guaranteed to produce at least $s+1$ elements, then it is guaranteed to produce at least s elements as well. In the case of inductive types, the situation is the opposite: size annotations occur positively, e.g. $\mathsf{list}^s\, T \leq \mathsf{list}^{\widehat{s}}\, T$. Recall that for inductive types, size annotations indicate an *upper bound* on the size of its elements.

The typing rules are given in Fig. 4. Most of the rules are standard. In rule (case), the argument M must have a successor type, i.e. it must guarantee to produce at least one element; otherwise, evaluation could get stuck. Note, in rule (cofix), that types valid for corecursion have the form $\Pi\Delta.\mathsf{stream}^{\imath}\, U$ where $\imath$ occurs negatively in Δ and does not occur in U. For example, valid types include $\mathsf{stream}^{\imath}\, T$ and $\mathsf{stream}^{\imath}\, T \rightarrow \mathsf{stream}^{\imath}\, T \rightarrow \mathsf{stream}^{\imath}\, T$. Note that the body M ensure that at least one more element is produced (with the return type $T\,[\widehat{\imath}/\imath]$).

$$\frac{\mathsf{simple}(T) \qquad \mathsf{simple}(U)}{\mathsf{simple}(\Pi x{:}T.U)} \qquad \frac{\mathrm{SV}(T) = \emptyset}{\mathsf{simple}(T)}$$

$$\frac{\mathsf{simple}(T)}{\mathsf{simple}(\mathsf{stream}^s\, T)}$$

Fig. 3. Simple types

Note the use of function $\mathrm{SV}(\cdot)$ in rules (abs), (app), (cons), (case), and (cofix). We add this conditions to restrict the occurrences of size variables in types, which is necessary for defining the Λ-set model.

Basically, we only allow *simple types* in the typing judgment. A type T is simple if it satisfies the predicate $\mathsf{simple}(T)$, defined by the rules of Fig. 3. By using $\mathrm{SV}(\cdot)$ in the above mentioned rules, we ensure that, in a typing judgment, all terms in type positions satisfy the simple predicate. That is, given a valid judgment $\Gamma \vdash M : T$, the following holds:

- $\mathsf{simple}(\Gamma)$ and $\mathsf{simple}(T)$;
- if $T \approx u$, for some sort u, then $\mathsf{simple}(M)$;
- if $T \not\approx u$, for all sorts u, then $\mathrm{SV}(M) = \emptyset$.

This property, easily proved by induction on the type derivation, is essential in our relational model, since only simple types have a relational interpretation (Sect. IV).

Restricting to simple types disallows, for example, types of the form $F\,(\mathsf{stream}^{\imath}\, T)$. However, in [5] we showed that such types have little use in systems with implicit size quantification like CC$\widehat{\omega}$. Basically, every size variable can be replaced by ∞ in a judgment, i.e. $\Gamma \vdash M : T$ implies $\Gamma^\infty \vdash M^\infty : T^\infty$. In a type derivation, the only needed size variables come from the introduction of cofixpoints. However, cofixpoint types have a precise form, $\Pi\Delta.\mathsf{stream}^{\imath}\, U$, which in general satisfies the simple predicate. Therefore, restricting to simple types does not pose a limitation in practice.

Metatheory: The metatheory of coinduction in the presence of dependent types is far from trivial. One particular issue is subject reduction: it is not valid for ν-reduction. This was observed by Giménez [11], who studied and implemented guarded coinduction in the Coq kernel. The unrestricted ν-reduction, denoted ν_{u},

$$(\mathsf{cofix}\ f{:}T^\star := M)\ \ \nu_{\mathsf{u}}\ \ (M\,[(\mathsf{cofix}\ f{:}T^\star := M)/f])$$

does satisfy SR, but it is not strongly normalizing. (We write $\rightarrow_{\mathsf{u}}$ for the compatible closure of $\beta\imath\nu_{\mathsf{u}}$-reduction.)

To get around this issue, we define two systems: in CC$\widehat{\omega}$, evaluation of programs and conversion use the restricted

reduction $\to$; in $\mathsf{uCC}\widehat{\omega}$, evaluation of programs is defined using $\to$, while the conversion rule is defined using $\to_{\mathsf{u}}$.

Since $\to \subseteq \to_{\mathsf{u}}$, it is clear that any program typable in $\mathsf{CC}\widehat{\omega}$ is also typable in $\mathsf{uCC}\widehat{\omega}$. Furthermore, strong normalization of $\mathsf{CC}\widehat{\omega}$ is a direct consequence of strong normalization of $\mathsf{uCC}\widehat{\omega}$. A corollary of this property is that conversion (and therefore, type checking) is decidable in $\mathsf{CC}\widehat{\omega}$. Hence, $\mathsf{uCC}\widehat{\omega}$ satisfies SR but type checking is not decidable (see [12] for a proof), while $\mathsf{CC}\widehat{\omega}$ is decidable but does not satisfy SR.

The Coq kernel implements ν-reduction, which does not satisfy SR, on the idea that having a terminating type-checking procedure is more important than having SR. Although it works relatively well in practice, this situation is not completely satisfactory.

We do not provide a definite solution to this problem in this paper. Instead, we focus on establishing soundness for a productivity criterion that greatly improves over guard predicates and can be readily implemented in the Coq kernel.

Research on this topic is very active. Although, as far as we know, no complete solution has been proposed yet, there are some promising approaches (e.g. [12], [13]).

The following lemma states some metatheoretical properties of $\mathsf{uCC}\widehat{\omega}$: weakening, substitution, uniqueness of types (up to erasure), and subject reduction:

Lemma 2.
- *If* $\Gamma \vdash M : T$, *then* $\Gamma\Delta \vdash M : T$.
- *If* $\Gamma_1(x : T)\Gamma_2 \vdash M : U$, $\Gamma_1 \vdash N : T$, *and* $\mathrm{SV}(N) = \emptyset$, *then* $\Gamma_1(\Gamma_2\,[N/x]) \vdash M\,[N/x] : U\,[N/x]$.
- *If* $\Gamma \vdash M : T_1$ *and* $\Gamma \vdash M : T_2$, *then* $|T_1| \approx |T_2|$.
- *If* $\Gamma \vdash M : T$ *and* $M \to M'$ *then* $\Gamma \vdash M' : T$

A. Examples

We consider some programs in $\mathsf{CC}\widehat{\omega}$ that show the advantages of type-based productivity against guard predicates as used in Coq. (We assume a signature containing a definition of natural numbers as well as some usual operations.)

A typical function on streams is map, defined as follows:

$$\begin{aligned}&\mathsf{map} : \Pi(A\,B{:}\mathsf{Type}_0).(A \to B) \to \mathsf{stream}^s\,A \to \mathsf{stream}^s\,B\\ &\mathsf{map} \stackrel{\text{def}}{=} \lambda A\,B\,f.\ \mathsf{cofix\ map} : \mathsf{stream}^\star\,A \to \mathsf{stream}^\star\,B :=\\ &\qquad \lambda x.\ \mathsf{case}\ x\ \mathsf{of}\ \mathsf{cons}(h,t) \Rightarrow \mathsf{cons}(f\,h, \mathsf{map}\,t)\end{aligned}$$

While this function is guarded (hence accepted by Coq), the advantage of $\mathsf{CC}\widehat{\omega}$, is that we can give it the size-preserving type $\mathsf{stream}^s\,A \to \mathsf{stream}^s\,B$ (note the use of position types to mark recursive positions).

Other examples of size-preserving functions are sum, that returns the pair-wise addition of two streams of natural numbers, and merge, that combines two ordered streams (we write Snat^s to mean $\mathsf{stream}^s\,\mathsf{nat}$):

$$\begin{aligned}&\mathsf{sum} : \mathsf{Snat}^s \to \mathsf{Snat}^s \to \mathsf{Snat}^s\\ &\mathsf{sum} \stackrel{\text{def}}{=} \mathsf{cofix\ sum} : \mathsf{Snat}^\star \to \mathsf{Snat}^\star \to \mathsf{Snat}^\star :=\\ &\qquad \lambda x_1\,x_2.\ \mathsf{case}\ x_1\ \mathsf{of}\ \mathsf{cons}(h_1,t_1) \Rightarrow\\ &\qquad\qquad \mathsf{case}\ x_2\ \mathsf{of}\ \mathsf{cons}(h_2,t_2) \Rightarrow\\ &\qquad\qquad\ \mathsf{cons}(h_1+h_2, \mathsf{sum}\,t_1\,t_2)\end{aligned}$$

$$\frac{}{\mathsf{WF}(\cdot)}\ (\mathsf{wf\text{-}emp}) \qquad \frac{\mathsf{WF}(\Gamma) \qquad \Gamma \vdash T : u}{\mathsf{WF}(\Gamma(x{:}T))}\ (\mathsf{wf\text{-}cons})$$

$$\frac{\mathsf{WF}(\Gamma) \qquad \Gamma(x) = T}{\Gamma \vdash x : T}\ (\mathsf{var})$$

$$\frac{\mathsf{WF}(\Gamma) \qquad (u_1,u_2) \in \mathsf{Axiom}}{\Gamma \vdash u_1 : u_2}\ (\mathsf{sort})$$

$$\frac{\Gamma \vdash T : u_1 \qquad \Gamma(x{:}T) \vdash U : u_2 \qquad (u_1,u_2,u_3) \in \mathsf{Rule}}{\Gamma \vdash \Pi x : T.U : u_3}\ (\mathsf{prod})$$

$$\frac{\Gamma(x{:}T) \vdash M : U \qquad \mathrm{SV}(M) = \emptyset}{\Gamma \vdash \lambda x : |T|.M : \Pi x : T.U}\ (\mathsf{abs})$$

$$\frac{\Gamma \vdash M : \Pi x : T.U \qquad \Gamma \vdash N : T \qquad \mathrm{SV}(N) = \emptyset}{\Gamma \vdash M\,N : U\,[N/x]}\ (\mathsf{app})$$

$$\frac{\Gamma \vdash M : T \qquad \Gamma \vdash U : u \qquad T \le U}{\Gamma \vdash M : U}\ (\mathsf{conv})$$

$$\frac{\Gamma \vdash T : \mathsf{Type}_0}{\Gamma \vdash \mathsf{stream}^s\,T : \mathsf{Type}_0}\ (\mathsf{stream})$$

$$\frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : \mathsf{stream}^s\,T \qquad \mathrm{SV}(M) = \emptyset}{\Gamma \vdash \mathsf{cons}(M,N) : \mathsf{stream}^{\widehat{s}}\,T}\ (\mathsf{cons})$$

$$\frac{\begin{array}{c}\Gamma \vdash M : \mathsf{stream}^{\widehat{s}}\,T \qquad \Gamma, x{:}\mathsf{stream}^{\widehat{s}}\,T \vdash P : u\\ \Gamma(y_1{:}T)(y_2{:}\mathsf{stream}^s\,T) \vdash N : P\,[\mathsf{cons}(y_1,y_2)/x]\\ \mathrm{SV}(N) = \emptyset\end{array}}{\Gamma \vdash \left(\begin{array}{l}\mathsf{case}_{|P|}\ x := M\ \mathsf{of}\\ \mathsf{cons}(y_1,y_2) \Rightarrow N\end{array}\right) : P\,[M/x]}\ (\mathsf{case})$$

$$\frac{\begin{array}{c}T \equiv \Pi\Delta.\mathsf{stream}^\imath\,U \qquad \imath\ \mathsf{neg}\ \Delta \qquad \imath \notin \mathrm{SV}(\Gamma,U,M)\\ \Gamma \vdash T : u \qquad \Gamma(f : T) \vdash M : T\,[\widehat{\imath}/\imath] \qquad \mathrm{SV}(M) = \emptyset\end{array}}{\Gamma \vdash \mathsf{cofix}\ f{:}|T|^\imath := M : T\,[s/\imath]}\ (\mathsf{cofix})$$

Fig. 4. Typing rules

$$\begin{aligned}&\mathsf{merge} : \mathsf{Snat}^s \to \mathsf{Snat}^s \to \mathsf{Snat}^s\\ &\mathsf{merge} \stackrel{\text{def}}{=} \mathsf{cofix\ merge} : \mathsf{Snat}^\star \to \mathsf{Snat}^\star \to \mathsf{Snat}^\star :=\\ &\qquad \lambda x_1\,x_2.\ \mathsf{case}\ x_1\ \mathsf{of}\ \mathsf{cons}(h_1,t_1) \Rightarrow\\ &\qquad\qquad \mathsf{case}\ x_2\ \mathsf{of}\ \mathsf{cons}(h_2,t_2) \Rightarrow\\ &\qquad\qquad\ \mathsf{if}\ h_1 \le h_2\ \mathsf{then}\ \mathsf{cons}(h_1, \mathsf{merge}\,t_1\,x_2)\\ &\qquad\qquad\ \mathsf{else}\ \mathsf{cons}(h_2, \mathsf{merge}\,x_1\,t_2)\end{aligned}$$

The following definitions are not guarded, but are well typed in $\mathsf{CC}\widehat{\omega}$, thanks to the size-preserving type of sum, merge, and map. The stream fib computes the Fibonacci sequence, while ham computes the streams of Hamming numbers:

$$\begin{aligned}&\mathsf{fib} \stackrel{\text{def}}{=} \mathsf{cofix\ fib} : \mathsf{Snat}^\star := \mathsf{cons}(0, \mathsf{sum\,fib\,(cons}(1, \mathsf{fib})))\\ &\mathsf{ham} \stackrel{\text{def}}{=} \mathsf{cofix\ ham} : \mathsf{Snat}^\star :=\\ &\qquad \mathsf{cons}(1, \mathsf{merge\,(map}\,(\lambda x.2*x)\,\mathsf{ham})\\ &\qquad\qquad (\mathsf{merge\,(map}\,(\lambda x.3*x)\,\mathsf{ham})\\ &\qquad\qquad\qquad (\mathsf{map}\,(\lambda x.5*x)\,\mathsf{ham})))\end{aligned}$$

However, the following alternative definition of fib is not well typed in CC$\widehat{\omega}$, even though it is productive:

$$\mathsf{fib}' \stackrel{\text{def}}{=} \mathsf{cofix}\ \mathsf{fib}' : \mathsf{Snat}^\star := \mathsf{cons}(0, (\mathsf{cons}(1, \mathsf{sum}\,\mathsf{fib}'\,(\mathsf{tail}\,\mathsf{fib}'))))$$

where the usual definition of tail has type $\mathsf{stream}^{\widehat{s}}\ A \to \mathsf{stream}^s\ A$. The problem is that $\mathsf{tail}\,\mathsf{fib}'$ is not well typed, since fib' does not have a successor stage in its type. In general, we cannot make corecursive calls to the result of an application of tail. (Although fib' can be written using mutual recursion.)

The function even filters out the odd positions of a stream:

$$\begin{aligned}&\mathsf{even} : \mathsf{Snat}^\infty \to \mathsf{Snat}^s\\ &\mathsf{even} \stackrel{\text{def}}{=} \mathsf{cofix}\ \mathsf{even} : \mathsf{Snat} \to \mathsf{Snat}^\star :=\\ &\qquad \lambda x.\ \mathsf{case}\ x\ \mathsf{of}\ \mathsf{cons}(h_1, t_1) \Rightarrow\\ &\qquad\quad \mathsf{case}\ t_1\ \mathsf{of}\ \mathsf{cons}(h_2, t_2) \Rightarrow \mathsf{cons}(h_2, \mathsf{even}\,t_2)\end{aligned}$$

However, our type system is not expressive enough to give even the more precise type $\mathsf{stream}^{\imath+\imath}\ A \to \mathsf{stream}^\imath\ A$, since our size algebra does not feature addition. This implies that, similarly to the case of tail, we cannot make corecursive calls to the result of an application of even.

IV. Strong Normalization

In this section we define a model for uCC$\widehat{\omega}$ based on Λ-sets [8], [14]. We establish strong normalization (SN) and logical consistency as a consequence of soundness.

In previous work [5], [7] we defined a Λ-set model for an extension of CCω with inductive types and a type-based mechanism for ensuring termination. To deal with sized types, we extended the Λ-set model with a relational interpretation.

To focus on the adaptation of our relational model to streams, we only prove SN for the predicative fragment of uCC$\widehat{\omega}$. That is, in this section, we only consider the fragment of uCC$\widehat{\omega}$ obtained by removing Prop.

Dealing with impredicativity in dependent type theories is a complicated issue. However, in the Λ-set model, the technical difficulties have been solved by Altenkirch [8], and Melliès and Werner [14]. In [5], we used Λ-sets to define a model of CIC with an impredicative universe and full inductive types.

Impredicativity is rather orthogonal to coinduction in our setting, since coinductive types are defined in the predicative hierarchy. In particular, corecursive definitions inhabit the predicate universes. Therefore, we restrict to the predicative fragment of CC$\widehat{\omega}$ to avoid obscuring the proof of SN with technical details that are not related to coinduction.

Overview of the proof: The Λ-set model combines a set-theoretical model with a realizability interpretation. A Λ-set X contains two main components: a carrier set $X_\circ$ that defines the set-theoretical interpretation, and a realizability relation $\models \subseteq \mathrm{SN} \times X_\circ$ that is used to prove strong normalization (SN is the set of strongly normalizing terms).

Terms are interpreted as elements in the carrier set of their types. Hence, soundness of the model is expressed as follows: if $\Gamma \vdash M : T$, then $[M](\gamma) \in [T](\gamma)$, where $[\cdot]$ is the interpretation function and γ the interpretation of Γ. SN follows from the fact that every term realizes its interpretation, i.e. $M \models [M](\gamma)$.

In order to deal with sized (inductive) types, in [5], [7], we extended the Λ-set model with a relational interpretation, denoted $[\![\cdot]\!]$, that takes size information into account. In this work we extend the relational interpretation to streams, although the method applies to coinductive types in general.

In the traditional interpretation, $[\cdot]$, we interpret streams by their set-theoretical counterpart, concretely finite and infinite sequences. We write $\langle \alpha_i \rangle_i$ to denote sequences (finite and infinite). We write π_i for the partial function returning the i-th element of a sequence (if defined).

The relational interpretation of streams is natural: the type $\mathsf{stream}^s\ T$ is interpreted as the set of pairs of (set-theoretical) streams, whose length is *at least* s, and whose elements are pairwise related by the interpretation of T:

$$[\![\mathsf{stream}^s\ T]\!] = \{(\alpha, \beta) : i < [s] \Rightarrow (\pi_i(\alpha), \pi_i(\beta)) \in [\![T]\!]\}$$

If s is ∞, the relational interpretation is the identity relation.

Let us illustrate the proof of soundness with a simplified cofixpoint definition. Consider the following typing rule:

$$\frac{(f : \mathsf{stream}^\imath\ T) \vdash M : \mathsf{stream}^{\widehat{\imath}}\ T}{\vdash \mathsf{cofix}\ f{:}\mathsf{stream}^\star\ T := M : \mathsf{stream}^s\ T}$$

Soundness of the relational interpretation states that given two streams (α_1, α_2) that coincide in the first $\imath$ positions, $([M](\alpha_1), [M](\alpha_2))$ is a pair of streams that coincide in the first $\imath + 1$ positions. Intuitively, starting from any pair of streams, and iterating $[M]$ ω-times, we obtain two streams that coincide everywhere.

In essence, the cofixpoint rule defines a *contractive function* in the relational model. Then, there is a unique fixed point: a stream that is invariant under cofixpoint unfolding. This unique stream is used as the interpretation of $\mathsf{cofix}\ f{:}\mathsf{stream}^\star\ T := M$.

The relational interpretation only makes sense for types. This is where the restriction to simple types comes into play: we define the relational interpretation by induction on the simple predicate. The case of streams is given above. In the case of products, the interpretation is a set of pair of functions that take related elements in the domain into related elements in the codomain. Otherwise, the type has no size variables, and the interpretation is just the identity relation.

A. Preliminary definitions

We introduce the notion of saturated sets and Λ-sets. We define saturated sets in terms of elimination contexts.

Definition 3 (Elimination context)**.** *An elimination context is a term with a "hole", denoted by* $[]$*, that belongs to the following grammar:*

$$E[] ::= [] \mid E[]\,\mathcal{T} \mid \mathsf{case}_{\mathcal{T}^\circ}\ \mathcal{V} := E[]\ \mathsf{of}\ \mathsf{cons}(\mathcal{V}, \mathcal{V}) \Rightarrow \mathcal{T}$$

We write $E[M]$ for the term obtained by replacing the hole of $E[]$ with M.

We define weak-head reduction in terms of elimination contexts: $E[M] \to_{\mathrm{wh}} E[N] \iff M\ \beta\iota\mu\ N$. An *atomic term*

is a term in *weak-head normal form* (whnf), where reduction is stopped by a variable, i.e., a term is atomic if it is of the form $E[x]$. We use AT to denote the set of atomic terms.

Definition 4 (Saturated set)**.** *A set of terms $X \subseteq \mathrm{SN}$ is saturated iff it satisfies the following conditions:*

1) $\mathrm{AT} \cap \mathrm{SN} \subseteq X$*;*
2) *if $M \in \mathrm{SN}$ and $M \to_{\mathrm{wh}} M'$ and $M' \in X$, then $M \in X$.*

In other words, saturated sets contain all strongly normalizing atomic terms and are closed by weak-head expansion. The greatest saturated set is SN, and the smallest saturated set is $\overline{\mathrm{AT}}$, that is the closure of AT under weak-head expansion.

In the following we introduce the main tool of our proof, Λ-sets, as well as some operations.

Definition 5 (Λ-set)**.** *A Λ-set is a triple $X = (X_\circ, \models_X, \bot_X)$ where $X_\circ$ is a non-empty set, witnessed by $\bot_X \in X_\circ$, and $\models_X \subseteq \mathcal{T} \times X_\circ$.*

$X_\circ$ is the *carrier-set* and the elements of $X_\circ$ are called the *carriers* of X. The terms M such that $M \models_X \alpha$ for some $\alpha \in X_\circ$ are called the *realizers* of α. The element $\bot_X$ is called the *atomic element* of X. We write $\alpha \sqsubset X$ for $\alpha \in X_\circ$. A Λ-set X is included in a Λ-set Y, written $X \subseteq Y$, if $X_\circ \subseteq Y_\circ$, $\models_X \subseteq \models_Y$, and $\bot_X = \bot_Y$.

The notion of *saturated Λ-set* is necessary for proving SN.

Definition 6 (Saturated Λ-set)**.** *A Λ-set X is said to be* saturated *if*

1) *every realizer is strongly normalizable;*
2) *the atomic element $\bot_X$ is realized by any atomic strongly normalizable term;*
3) *for every $\alpha \in X_\circ$, if $N \models_X \alpha$, and $M \to_{\mathrm{wh}} N$ with $M \in \mathrm{SN}$, then $M \models_X \alpha$ (i.e., the realizers are closed under weak-head expansion).*

Note that the set of realizers of a saturated Λ-set is a saturated set. We define a product operation on Λ-sets that is used to interpret the product construction on terms.

Definition 7 (Product)**.** *Let X be a Λ-set and $\{Y_\alpha\}_{\alpha \in X_\circ}$ an $X_\circ$-indexed family of Λ-sets. We define the Λ-set $\Pi(X, Y)$ by:*

- $\Pi(X,Y)_\circ \stackrel{\text{def}}{=} \{f \in X_\circ \to \bigcup_{\alpha \in X_\circ} (Y_\alpha)_\circ : \forall \alpha \in X_\circ . f(\alpha) \in (Y_\alpha)_\circ\}$*;*
- $M \models_{\Pi(X,Y)} f \iff \forall \alpha \in X_\circ .\ N \models_X \alpha \Rightarrow M\,N \models_{Y_\alpha} f(\alpha)$
- $\bot_{\Pi(X,Y)} \stackrel{\text{def}}{=} \alpha \in X_\circ \mapsto \bot_{Y_\alpha}$.

Saturated Λ-sets are closed under products:

Lemma 8. *If X and every $\{Y_\alpha\}_{\alpha \in X_\circ}$ are saturated Λ-sets, so is $\Pi(X, Y)$.*

The following operation is useful for defining the relational interpretation of types.

Definition 9. *Let X be a Λ-set. Then X^2 is a Λ-set where*

- $(X^2)_\circ = \{(\alpha, \alpha) : \alpha \sqsubset X\}$
- $M \models_{X^2} (\alpha, \alpha) \iff M \models_X \alpha$
- $\bot_{X^2} = (\bot_X, \bot_X)$

B. The model

We define two interpretations: the first, the term interpretation, does not take size annotations into account; the second, the relational interpretation, only makes sense for simple types and uses size annotations.

We require the use of inaccessible cardinals to interpret the predicative hierarchy of universes. This is common in models of type theory (e.g. [15], [16]). We assume an increasing sequence of inaccessible cardinals $\{\lambda_i\}_{i \in \mathbb{N}}$.

We define $\mathcal{U}_i$ to be the set of saturated Λ-sets whose carrier-set are in V_{λ_i}—where V_α represents the cumulative hierarchy of sets. The set $\mathcal{U}_i$ can itself be viewed as the Λ-set $(\mathcal{U}_i, \mathrm{SN} \times \mathcal{U}_i, \{\emptyset\})$. Note that $\mathcal{U}_i \in \mathcal{U}_{i+1}$ which verifies the typing rule $\mathsf{Type}_i : \mathsf{Type}_{i+1}$.

Term interpretation: We interpret streams by finite and infinite sequences of elements in the interpretation of the type. To define the realizers, we need to force reduction of a term by placing it inside a case analysis. We define $\downarrow M \stackrel{\text{def}}{=} \mathsf{case}_{\mathsf{Type}_0}\ x := M\ \mathsf{of}\ \mathsf{cons}(y_1, y_2) \Rightarrow \mathsf{cons}(y_1, y_2)$.

Given a Λ-set X, we define $\mathcal{S}(X)$ as the Λ-set given by

- $\mathcal{S}(X)_\circ \stackrel{\text{def}}{=} \{(\emptyset)\} \cup \{(\alpha_0, \ldots, \alpha_i, \emptyset) : i \geq 0 \wedge \alpha_i \sqsubset X\} \cup \{(\alpha_0, \alpha_1, \ldots) : \alpha_i \sqsubset X\}$
- $M \models_{\mathcal{S}(X)} \beta$ with $M \in \mathrm{SN}$, iff
  - $\beta = (\emptyset)$, and $\downarrow M \to^*_{\mathrm{wh}} N \in \mathrm{AT}$;
  - $\beta = \langle \alpha_i \rangle_{i=0\ldots}$ and $\downarrow M \to^*_{\mathrm{wh}} \mathsf{cons}(N, P)$ where $N \models_X \alpha_0$, and $P \models_{\mathcal{S}(X)} \langle \alpha_i \rangle_{i=1\ldots}$
- $\bot \stackrel{\text{def}}{=} (\emptyset)$

We define the interpretation function by induction on the structure of terms. We write $[\Gamma \vdash M]_\gamma$ for the interpretation of M under the interpretation γ of Γ. We use Hilbert's choice operator, ϵ, to define the interpretation of cofixpoints to ensure that the interpretation is invariant under (relaxed) ν-reductions.

Definition 10 (Interpretation of terms and contexts)**.**

- $[\cdot] = \{\emptyset\}$
- $[\Gamma(x{:}T)] = \{(\gamma, \alpha) : \gamma \in [\Gamma] \wedge \alpha \sqsubset [\Gamma \vdash T]_\gamma\}$
- $[\Gamma \vdash \mathsf{Type}_i]_\gamma = \mathcal{U}_i$
- $[\Gamma \vdash x]_\gamma = \gamma(x)$
- $[\Gamma \vdash \Pi x : T.U]_\gamma = \Pi([\Gamma \vdash T]_\gamma, [\Gamma(x : T) \vdash U]_{\gamma,\_})$
- $[\Gamma \vdash \lambda x{:}T^\circ.M]_\gamma = [\Gamma(x : T) \vdash M]_{\gamma,\_}$
- $[\Gamma \vdash M\,N]_\gamma = ([\Gamma \vdash M]_\gamma)([\Gamma \vdash N]_\gamma)$
- $[\Gamma \vdash \mathsf{stream}^\infty\, T]_\gamma = \mathcal{S}([\Gamma \vdash T]_\gamma)$
- $[\Gamma \vdash \mathsf{cons}(M, N)]_\gamma = ([\Gamma \vdash M]_\gamma, [\Gamma \vdash N]_\gamma)$
- $[\Gamma \vdash \mathsf{case}_{P^\circ}\ x := M\ \mathsf{of}\ \mathsf{cons}(y_1, y_2) \Rightarrow N]_\gamma =$
  - $[\Gamma(y_1 : T)(y_2 : \mathsf{stream}\, T) \vdash N]_{\gamma, \alpha_0, \langle \alpha_i \rangle_{i=1\ldots}}$, *if* $[\Gamma \vdash M]_\gamma = \langle \alpha_i \rangle_{i=0\ldots}$
  - $\bot_{[\Gamma\,(x:\mathsf{stream}\,T) \vdash P](\gamma, (\emptyset))}$, *if* $[\Gamma \vdash M]_\gamma = (\emptyset)$
- $[\Gamma \vdash \mathsf{cofix}\ f{:}T^\star := M]_\gamma = \epsilon(F, P)$*, where* $F \sqsubset [\Gamma \vdash T]_\gamma$*, and $P(F)$ is the following property:*

$$F = [\Gamma(f : |T|) \vdash M]_{\gamma, F}$$

We write $[\Gamma(x{:}T) \vdash M]_{\gamma,\_}$ as a short hand for $\delta \sqsubset [\Gamma \vdash T]_\gamma \mapsto [\Gamma(x{:}T) \vdash M]_{\gamma,\delta}$.

The above interpretation is sound with respect to weakening, substitution, and subject reduction as stated in the following lemma.

Lemma 11.
- *(Weakening)* $[\Gamma(z:T^\circ)\Delta \vdash M]_{\gamma,\alpha,\delta} \doteq [\Gamma\Delta \vdash M]_{\gamma,\delta}$.
- *(Substitution) Let* $\mathrm{SV}(N) = \emptyset$ *and* $\nu = [\Gamma \vdash N]_\gamma$. *Then,* $[\Gamma, \Delta\,[N/x] \vdash M\,[N/x]]_{\gamma,\delta} \doteq [\Gamma(x:T)\Delta \vdash M]_{\gamma,\nu,\delta}$.
- *(Reduction) If* $M \to_{\mathsf{u}} N$, *then* $[\Gamma \vdash M]_\gamma = [\Gamma \vdash N]_\gamma$.

Proving that ν-reduction is invariant is trivial thanks to the use of Hilbert's choice operator.

The relational interpretation: The relational interpretation of types (RI) is used to ensure that terms respect size information given by their types. We write $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2}$ for the RI of T under interpretations γ_1 and γ_2 of Γ, and a stage interpretation π (defined below). The RI of a type is a Λ-set whose carrier set is composed of pairs.

Definition 12 (Relational type)**.** *Let* X_1 *and* X_2 *be* Λ*-sets. A* relational type for X_1 and X_2 *is a* Λ*-sets* X *such that* $X_\circ \subseteq X_{1\circ} \times X_{2\circ}$, *and* $\bot_X = (\bot_{X_1}, \bot_{X_2})$. *We denote it by* $X \in \mathcal{R}(X_1, X_2)$.

We define the RI by induction on the structure of simple types. We need to define the interpretation of stages, products, and streams. Stages are interpreted by ordinals. In $\mathsf{uCC}\widehat{\omega}$, the greatest ordinal we need to consider is ω. However, if we add general coinductive types, we would need to consider greater ordinals (see [5] for the case of general inductive types).

Definition 13 (Stage interpretation)**.** *A stage assignment* π *is a function from* $\mathcal{V}_\mathcal{S}$ *to* $\mathfrak{a}$. *The interpretation of a stage* s *under* π, *written* $(\!|s|\!)_\pi$, *is defined by:*

$$(\!|\imath|\!)_\pi = \pi(\imath), \quad (\!|\infty|\!)_\pi = \omega, \quad (\!|\widehat{s}|\!)_\pi = \mathsf{min}((\!|s|\!)_\pi + 1, \omega)\ .$$

We define a relational interpretation for products. Let $\Pi(X_i, \alpha_i \sqsubset X_i \mapsto Y_{i,\alpha_i})$ for $i = 1, 2$ be two product spaces of Λ-sets. Let X be a Λ-set such that $X \in \mathcal{R}(X_1, X_2)$. For each $(\alpha_1, \alpha_2) \sqsubset X$, let Y_{α_1,α_2} be a Λ-set such that $Y(\alpha_1, \alpha_2) \in \mathcal{R}(Y_{1,\alpha_1}, Y_{2,\alpha_2})$. We define a relational type $\Pi^+(X, Y)$ for $\Pi(X_1, Y_1)$ and $\Pi(X_2, Y_2)$:

$$\Pi^+(X, Y) \in \mathcal{R}(\Pi(X_1, Y_1), \Pi(X_2, Y_2)),$$

by the following rules.

- $\Pi^+(X,Y)_\circ \stackrel{\mathrm{def}}{=} \{(f_1, f_2) \sqsubset \Pi(X_1, Y_1) \times \Pi(X_2, Y_2) \mid (\alpha_1, \alpha_2) \sqsubset X \Rightarrow (f_1\alpha_1, f_2\alpha_2) \sqsubset Y_{\alpha_1,\alpha_2}\}$;
- $M \models_{\Pi^+(X,Y)} (f_1, f_2) \iff \forall(\alpha_1, \alpha_2) \sqsubset X.\ N \models_X (\alpha_1, \alpha_2) \Rightarrow M\,N \models_{Y(\alpha_1,\alpha_2)} (f_1\alpha_1, f_2\alpha_2)$;
- $\bot_{\Pi^+(X,Y)} \stackrel{\mathrm{def}}{=} (\bot_{\Pi(X_1,Y_1)}, \bot_{\Pi(X_2,Y_2)})$

The carrier set of a relational product is formed by pairs of functions that take related elements in the domain into related elements in the codomain.

Finally, we define the relational interpretation of streams. Intuitively, two elements of $[\![\Gamma \vdash \mathsf{stream}^s\,T]\!]^\pi_{\gamma_1\sim\gamma_2}$ are related if the first $(\!|s|\!)_\pi$ elements are related in $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2}$. In particular, they must contain at least $(\!|s|\!)_\pi$ elements.

Given a relational Λ-set $X \in \mathcal{R}(X_1, X_2)$, we define $\mathcal{S}^\mathfrak{a}_R(X) = (X^\mathfrak{a}, \models^\mathfrak{a}, \bot^\mathfrak{a})$ by induction on $\mathfrak{a}$ as follows:

- if $\mathfrak{a} = 0$, then
  - $X^0 \stackrel{\mathrm{def}}{=} \mathcal{S}(X_1)_\circ \times \mathcal{S}(X_2)_\circ$,
  - $\models^0 \stackrel{\mathrm{def}}{=} \mathrm{SN} \times X^0$
  - $\bot^0 \stackrel{\mathrm{def}}{=} (\emptyset, \emptyset)$
- if $\mathfrak{a} = \mathfrak{b} + 1$, then
  - $X^\mathfrak{a} \stackrel{\mathrm{def}}{=} \{(\alpha, \alpha') \in \mathcal{S}(X_1)_\circ \times \mathcal{S}(X_2)_\circ : i < \mathfrak{a} \Rightarrow (\pi_i(\alpha), \pi_i(\alpha')) \sqsubset X\}$ where we require that $\pi_i(\alpha)$ and $\pi_i(\alpha')$ are both defined or both undefined.
  - $M \models^\mathfrak{a} (\alpha, \alpha')$ iff $M \in \mathrm{SN}$ and
    - $(\alpha, \alpha') = (\emptyset, \emptyset)$ implies $\downarrow M \to^*_{\mathrm{wh}} N \in \mathrm{AT}$;
    - $(\alpha, \alpha') = (\langle\alpha_i\rangle_{i=0\ldots}, \langle\alpha'_i\rangle_{i=0\ldots})$ implies $\downarrow M \to^*_{\mathrm{wh}} \mathsf{cons}(N_1, N_2)$, $N_1 \models_X (\alpha_0, \alpha'_0)$ and $N_2 \models^\mathfrak{b} (\langle\alpha_i\rangle_{i=1\ldots}, \langle\alpha'_i\rangle_{i=1\ldots})$
  - $\bot^\mathfrak{a} \stackrel{\mathrm{def}}{=} (\emptyset, \emptyset)$.
- if $\mathfrak{a} = \omega$, then $\mathcal{S}^\omega_R(X) \stackrel{\mathrm{def}}{=} \bigcap_{\mathfrak{a}<\omega} \mathcal{S}^\mathfrak{a}_R(X)$.

Now we are ready to define the relational interpretation. We proceed by induction on the definition of simple types. Let T be a type such that $\mathsf{simple}(T)$.

- If $T \equiv \Pi x : T_1.T_2$, we define $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2}$ as
$$\Pi^+([\![\Gamma \vdash T_1]\!]^\pi_{\gamma_1\sim\gamma_2}, [\![\Gamma(x:T_1) \vdash T_2]\!]^\pi_{\gamma_1,\_\sim\gamma_2,\_})$$
- If $T \equiv \mathsf{stream}^s\,T_1$, then $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2} \stackrel{\mathrm{def}}{=} \mathcal{S}^{(\!|s|\!)_\pi}_R([\![\Gamma \vdash T_1]\!]^\pi_{\gamma_1\sim\gamma_2})$.
- Otherwise, $\mathrm{SV}(T) = \emptyset$, and $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2} \stackrel{\mathrm{def}}{=} ([\Gamma \vdash T]_{\gamma_1})^2$.

As in the interpretation of terms, we extend the relational interpretation to contexts, by the following rules:

$$[\![\Gamma]\!]^\pi = \{(\emptyset, \emptyset\}$$
$$[\![\Gamma(x{:}T)]\!]^\pi = \{((\gamma_1, \alpha_1), (\gamma_2, \alpha_2)) : (\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi \wedge (\alpha_1, \alpha_2) \sqsubset [\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2}\}$$

The relational interpretation is sound with respect to weakening, substitution, subject reduction, and subtyping as stated in the following lemma.

Lemma 14.
- *(Weakening)* $[\![\Gamma\Delta \vdash U]\!]^\pi_{(\gamma_1,\delta_1)\sim(\gamma_2,\delta_2)} \doteq [\![\Gamma(z:T)\Delta \vdash U]\!]^\pi_{(\gamma_1,\alpha_1,\delta_1)\sim(\gamma_2,\alpha_2,\delta_2)}$
- *(Substitution) Let* $\mathrm{SV}(N) = \emptyset$ *and* $\nu_i = [\Gamma \vdash N]_{\gamma_i}$. *Then* $[\![\Gamma\Delta\,[N/x] \vdash T\,[N/x]]\!]^\pi_{(\gamma_1,\delta_1)\sim(\gamma_2,\delta_2)} \doteq [\![\Gamma(x:U)\Delta \vdash T]\!]^\pi_{(\gamma_1,\nu_1,\delta_1)\sim(\gamma_2,\nu_2,\delta_2)}$.
- *(Subject Reduction) If* $T \to_{\mathsf{u}} U$, *then* $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2} \doteq [\![\Gamma \vdash U]\!]^\pi_{\gamma_1\sim\gamma_2}$
- *(Subtyping) If* $T \leq U$, *then* $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2} \subseteq [\![\Gamma \vdash U]\!]^\pi_{\gamma_1\sim\gamma_2}$.

The following theorem states soundness of the Λ-set model. We write $[\Gamma \vdash M]_{\gamma_1,\gamma_2}$ for $([\Gamma \vdash M]_{\gamma_1}, [\Gamma \vdash M]_{\gamma_2})$.

Theorem 15 (Soundness of the Λ-set model)**.** *If $\Gamma \vdash M : T$, with $\mathrm{SV}(M) = \emptyset$, and $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$, then $[\Gamma \vdash M]_{\gamma_1,\gamma_2}$, $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1 \sim \gamma_2}$ are defined and*

$$[\Gamma \vdash M]_{\gamma_1,\gamma_2} \sqsubset [\![\Gamma \vdash T]\!]^\pi_{\gamma_1 \sim \gamma_2} \ .$$

Proof: By induction on the type derivation. The interesting case is rule (cofix), where M is $\mathsf{cofix}\ f{:}T^\star := M$. We need to prove that the use of Hilbert's choice operator is well defined, i.e. that there is exactly one function satisfying the required property. Intuitively, we prove that, starting from any two stream functions in the RI of T, and iterating the interpretation of M ω-times, we obtain two stream functions that coincide everywhere. (See Appendix B for more details). □

C. Strong Normalization

We can prove SN from Soundness of the Λ-set model, by showing that a term realizes its interpretation.

Definition 16. *Let $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$. We define $\theta \models^\Gamma_\pi (\gamma_1, \gamma_2)$ by the following clauses:*

$$\frac{}{\varepsilon \models^\Gamma_\pi \cdot} \qquad \frac{\theta \models^\Gamma_\pi (\gamma_1, \gamma_2) \qquad M \models_{[\![\Gamma \vdash T]\!](\gamma_1 \sim \gamma_2, \pi)} (\alpha_1, \alpha_2)}{\theta(x \mapsto M) \models^{\Gamma(x:T)}_\pi (\gamma_1, \alpha_1), (\gamma_2, \alpha_2)}$$

Lemma 17. *If $\Gamma \vdash M : T$ and $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$ and $\theta \models^\Gamma_\pi (\gamma_1, \gamma_2)$, then $M\theta \models_{[\![\Gamma \vdash T]\!](\gamma_1 \sim \gamma_2, \pi)} [\Gamma \vdash M]_{\gamma_1,\gamma_2}$*

Corollary 18. *If $\Gamma \vdash M : T$ then M is strongly normalizing.*

Proof: By showing that the identity relation, id, satisfies $\mathsf{id} \models^\Gamma_\pi \bot_\Gamma$, where $\bot_\Gamma$ is the sequence of atomic elements in the relational interpretation of the types of Γ. □

As a consequence of soundness and strong normalization, we can easily derive logical consistency.

Theorem 19. *There is no M such that $\cdot \vdash M : \Pi x : \mathsf{Type}_0.x$.*

V. Type Inference

A consequence of SN is decidability of type checking: conversion is decidable by reducing both types to normal form and comparing for α-equality. In this section, we adapt the size-inference algorithms for inductive types given in [6], [10] to our system.

For performing size inference, we consider a variant of CC$\widehat{\omega}$ where size variables are drawn from a set of *inference size variables* $\mathcal{V}_{\mathcal{S}_\mathcal{I}} = \{\alpha, \beta, \ldots\}$. The set of *generic terms* is the set of terms constructed using inference size variables. A generic term can be instantiated by replacing inference size variables with stage expressions, thus obtaining a sized term.

The algorithm works as follows: starting from a bare term M°, the algorithm computes a generic term M, a generic type T, and a minimal constraint set C on stage variables, such that every instantiation ρ that satisfies C, ρM has type ρT.

Formally, a *stage constraint* is a pair of inference stage expressions, written $s \sqsubseteq r$. A *constraint set* is a finite set of stage constraints. We say that a stage substitution $\rho : \mathcal{V}_{\mathcal{S}_\mathcal{I}} \to \mathcal{S}$ satisfies a constraint system C, written $\rho \models C$, if for every constraint $s_1 \sqsubseteq s_2$ in C, we have $\rho(s_1) \sqsubseteq \rho(s_2)$. Note that every constraint set is satisfiable by the substitution that replaces every variable with ∞.

The inference algorithm is defined by the following judgments:

- $C, \Gamma \vdash M^\circ \rightsquigarrow C', M \Rightarrow T$: given a context Γ, a bare term M°, and a set of constraints C, the algorithm either computes M satisfying $|M| \equiv M^\circ$, a type T, and a new set of constraints C', such that for all $\rho \models C'$, $\rho\Gamma \vdash \rho M' : \rho T$; or fails if no such M, T, and C' exists.
- $C, \Gamma \vdash M^\circ \Leftarrow T \rightsquigarrow C', M$: given a context Γ, a bare term M°, a type T and a set of constraints C, the algorithm either computes M satisfying $|M| \equiv M^\circ$ and a new set of constraints C', such that for all $\rho \models C'$, $\rho\Gamma \vdash \rho M' : \rho T$; or fails if no such M and C' exists.

The adaptation of the algorithms in [6], [10] to CC$\widehat{\omega}$ is relatively easy, since recursive and corecursive definitions have similar typing rules in the sized-types approach. We need to consider two features of our system: simple types and the implicit argument in cons. To ensure that a type satisfies the simple predicate we only need to add constraints that restrict the occurrences of size variables. For example, we ensure that no size variables occur in a type T by adding $\infty \sqsubseteq \mathrm{SV}(T)$.

To deal with the implicit argument of cons, we compute *least upper bounds* and *greatest lower bounds* on types. E.g., given T_1 and T_2, $T = T_1 \sqcup T_2$ satisfies the following property: for any substitution ρ and type T' such that $\rho T_1, \rho T_2 \leq T'$, we have $\rho T \leq T'$.

The type inference algorithm relies on a procedure called RecCheck that is used to check productivity of corecursive definitions, originally introduced in [10] for checking termination of recursive functions. Since the typing rules for fixpoint and cofixpoint are similar, we can use the same procedure in both cases.

Given a constraint system computed from the body of a corecursive definition, RecCheck verifies if the constraint set can be instantiated in a way that satisfies the conditions imposed by the (cofix) typing rule. Informally, $\mathsf{RecCheck}(\alpha, V^\star, C)$ succeeds (returning a constraint set) if α can be mapped to a fresh variable $\imath$, the variables in $V^\star$ can be mapped to stages with base $\imath$, and every other variable β in C is mapped to a stage not containing $\imath$.

Formally, RecCheck satisfies the following properties. We write $\rho \leq_{V_1} \rho'$ as a shorthand for $\rho(\alpha) \sqsubseteq \rho'(\alpha)$ for all $\alpha \in V_1$ and $\rho(\alpha) = \rho'(\alpha)$ for all $\alpha \notin V_1$.

- **Soundness (SRC):** if $\mathsf{RecCheck}(\alpha, V^\star, C') = C$ then for all ρ, such that $\rho \models C$, there exists a fresh stage variable $\imath$ and ρ' such that $\rho' \models C'$, $\rho'(\alpha) = \imath$, $[\rho(\alpha)/\imath]\,\rho' \leq_{V^\star} \rho$, and $\lfloor \rho'(V^\star) \rfloor = \imath$, and $\lfloor \rho'(\mathrm{SV}(C') \setminus V^\star) \rfloor \neq \imath$.
- **Completeness (CRC):** if $\rho(\alpha) = \imath$ and $\lfloor \rho(V^\star) \rfloor = \imath$ and $\lfloor \rho(\mathrm{SV}(C') \setminus V^\star) \rfloor \neq \imath$ and also $\rho \models C'$ then $\mathsf{RecCheck}(\alpha, V^\star, C')$ is defined and $\rho \models \mathsf{RecCheck}(\alpha, V^\star, C')$

The algorithm for computing RecCheck and the proofs of correctness can be found in [10].

In defining the algorithm, we use the following procedures:

- $T_1 \sqsubseteq T_2$ computes a constraint set C such that for all ρ, $\rho \models C$ iff $\rho T_1 \leq \rho T_2$;
- $T_1 \sqcup_C T_2 \rightsquigarrow T', C'$ (resp. $T_1 \sqcap_C T_2 \rightsquigarrow T', C'$) computes a type T' (fresh with respect to C) and a constraint set C' such that for all $\rho \models C$ and T such that $\rho T_1, \rho T_2 \leq T$ (resp. $T \leq \rho T_1, \rho T_2$) there exists $\rho' \models C'$ such that $\rho =_{\mathrm{SV}(C)} \rho'$ and $\rho' T' \leq T$ (resp. $T \leq \rho' T'$).

These procedures work by reducing T_1 and T_2 to normal form and extracting constraints from the size expressions in corresponding positions. They fail if the normal forms do not have the same shape, i.e. if $|\mathsf{nf}(T_1)| \not\equiv |\mathsf{nf}(T_2)|$.

The algorithm is defined in Fig. 5. We write $C, \Gamma \vdash M^\circ \rightsquigarrow C' \Rightarrow T$ for $C, \Gamma \vdash M^\circ \rightsquigarrow C', M \Rightarrow T$ when M is just $(M^\circ)^\infty$. And similarly for $C, \Gamma \vdash M^\circ \Leftarrow T \rightsquigarrow C'$. We write $C, \Gamma \vdash M^\circ \rightsquigarrow C' \Rightarrow^* W$ as a shorthand for $C, \Gamma \vdash M^\circ \rightsquigarrow C' \Rightarrow T \wedge \mathsf{whnf}(T) = W$.

Let us briefly explain the rules related to streams. In rule (a-stream) we check that the argument to stream is a type and introduce a fresh size variable in the result. In rule (a-cons) we independently check both arguments to cons and then take the join (least upper bound) of the result. In rule (a-case) we first check the argument M°, then we check the return type adding the constraint $\widehat{\alpha} \sqsubseteq r$ to ensure that the type of M has a successor size; finally, we check the type of the only branch. In rule (a-cofix) first we check the type of the cofixpoint, and then its body; we call RecCheck to ensure the condition of typing rule (cofix) are satisfied, adding the constraint $\widehat{\Delta} \sqsubseteq \Delta$ to ensure that α occurs negatively in Δ.

The type-inference algorithm is sound and complete:

Lemma 20 (Soundness of type inference)**.**

1) *If* $C, \Gamma \vdash M^\circ \Leftarrow T \rightsquigarrow C', M$*, then for all* $\rho \models C'$*,* $\rho\Gamma \vdash \rho M : \rho T$*.*
2) *If* $C, \Gamma \vdash M^\circ \rightsquigarrow C', M \Rightarrow T$*, then for all* $\rho \models C'$*,* $\rho\Gamma \vdash \rho M : \rho T$*.*

Lemma 21 (Completeness of type inference)**.**

1) *If* $\rho\Gamma \vdash M : \rho T$*,* $\rho \models C$*, and* $\mathrm{SV}(\Gamma, T) \subseteq V$*, then there exist* C', M', ρ' *such that* $\rho' \models C'$*,* $\rho =_V \rho'$*,* $\rho' M' = M$*, and* $C, \Gamma \vdash |M| \Leftarrow T \rightsquigarrow C', M'$
2) *If* $\rho\Gamma \vdash M : T$*,* $\rho \models C$*, and* $\mathrm{SV}(\Gamma) \subseteq V$*, there exist* C', M', T', ρ' *such that* $\rho' \models C'$*,* $\rho' T' \leq T$*,* $\rho =_V \rho$*,* $\rho' M' = M$*, and* $C, \Gamma \vdash |M| \rightsquigarrow C', M' \Rightarrow T'$*.*

VI. Related Work

Guarded recursion: The use of guarded definitions to ensure productivity in type theory was introduced by Coquand [17]. Giménez proves SN for an extension of CC with streams and guarded recursion [11]. He already observed that subject reduction (SR) is not valid.

Finding a satisfactory representation of coinduction in type theory that does not have such problems with SR is currently an important question in Type Theory. McBride [12] proposes to use Observational Type Theory. Abel et al. [13] propose the use of *copatterns* to define corecursive functions. In a nutshell, copatterns allows to define corecursive functions by their observations. This approach satisfies SR; however, the authors only handle the simply-typed case, and in particular they do not address the issue of productivity (only coverage).

Altenkirch and Danielsson [18] describe coinduction in Agda, which allows mixing induction and coinduction. Reduction is based on operators that delay and force evaluation of a corecursive value. This approach does not suffer from the SR problem, since dependent elimination is not allowed.

Type-based approaches: The literature on termination and productivity checking using sized types is too vast to review here—see e.g. [4], [5]. We focus on work related to dependent type theories.

In [19], [20], Abel proposes extensions of dependent type theories with sized types and first-class sizes. While this approach is more expressive than ours, it imposes a greater burden on the user, as sizes are not hidden—although the task could be simplified by inferring implicit size information. An experimental version of Agda incorporates these extensions. However, as far as we know, there is no comprehensive metatheoretical study.

In [5] we sketched some of the ideas presented in this paper.

Other approaches: Severi and de Vries [21] propose an extension of Pure Type Systems with streams, called coPTS. Productivity is ensured by the type system by keeping track of the *depth* of a stream. They use a judgment of the form $\Gamma \vdash_i M : T$ that types M at a depth i. Productivity is ensured by a rule of the form:

$$\frac{\Gamma(f :_{i+1} T) \vdash_i M : T}{\Gamma \vdash_i \mathsf{cofix}\ f{:}T := M : T}$$

coPTS allow to type stream definitions that use tail (such as fib′), but not functions like even. However, this approach is less compositional than sized types, since size information is not exported in the type of a function (but tied to the judgment).

Hutton and Jaskelliof [22] propose the use of contractive functions as a way to ensure productivity of stream definitions in Haskell. They show that any contractive function $\mathsf{stream}\, T \to \mathsf{stream}\, T$ can be generated from a function of type $\mathsf{list}\, T \to T$, and viceversa. However, writing functions this way is not always simple (e.g. the Hamming sequence).

Semantics based on contractive functions have been proposed several times in the literature, e.g. [23], [24] and, in a more general setting, in [25].

VII. Conclusions and Future Work

We presented an extension of the Calculus of Constructions with a type-based mechanism for ensuring productivity of stream definitions. We have proved strong normalization and logical consistency based on a Λ-set model. We also presented a size-inference algorithm, inspired from CIC$\hat{}$.

While deep theoretical issues remain, this approach presents an important improvement over guard predicates currently implemented in Coq. It is more expressive and easier to understand, while posing little burden on the user. Furthermore, it allows to treat induction and coinduction in a uniform way, both at the theoretical and practical level.

$$\frac{}{C,\Gamma \vdash u \rightsquigarrow C \Rightarrow \mathsf{Axiom}(u)}\ \textsf{(a-sort)} \qquad \frac{}{C,\Gamma \vdash x \rightsquigarrow C \Rightarrow \Gamma(x)}\ \textsf{(a-var)} \qquad \frac{C,\Gamma \vdash M^\circ \rightsquigarrow C' \Rightarrow T'}{C,\Gamma \vdash M^\circ \Leftarrow T \rightsquigarrow C' \cup T' \leq T}\ \textsf{(a-check)}$$

$$\frac{\begin{array}{c} C,\Gamma \vdash {T_1}^\circ \rightsquigarrow C_1, T_1 \Rightarrow^* u_1 \\ C_1,\Gamma(x:T_1) \vdash {M}^\circ \rightsquigarrow C_2 \Rightarrow T_2 \qquad C_3 = C_2 \cup \infty \sqsubseteq \mathrm{SV}(M) \end{array}}{C,\Gamma \vdash \lambda x : {T_1}^\circ . {M}^\circ \rightsquigarrow C_3 \Rightarrow \Pi x : T_1 . T_2}\ \textsf{(a-abs)}$$

$$\frac{\begin{array}{c} C,\Gamma \vdash {T_1}^\circ \rightsquigarrow C_1, T_1 \Rightarrow^* u_1 \\ C_1,\Gamma(x:T_1) \vdash {T_2}^\circ \rightsquigarrow C_2, T_2 \Rightarrow^* u_2 \end{array}}{C,\Gamma \vdash \Pi x : {T_1}^\circ . {T_2}^\circ \rightsquigarrow C_2, \Pi x : T_1 . T_2 \Rightarrow \mathsf{Rule}(u_1,u_2)}\ \textsf{(a-prod)} \qquad \frac{\begin{array}{c} C,\Gamma \vdash {M_1}^\circ \rightsquigarrow C_1 \Rightarrow^* \Pi x : T_2 . T \\ C_1,\Gamma \vdash {M_2}^\circ \Leftarrow T_2 \rightsquigarrow C_2 \\ C_3 = C_2 \cup \infty \sqsubseteq \mathrm{SV}(M_2) \end{array}}{C,\Gamma \vdash {M_1}^\circ\, {M_2}^\circ \rightsquigarrow C_3 \Rightarrow T\,[M_2/x]}\ \textsf{(a-app)}$$

$$\frac{C,\Gamma \vdash T^\circ \Leftarrow \mathsf{Type}_0 \rightsquigarrow C_1, T \qquad \alpha \text{ fresh}}{C,\Gamma \vdash \mathsf{stream}\, T^\circ \rightsquigarrow C_1, \mathsf{stream}^\alpha\, T \Rightarrow \mathsf{Type}_0}\ \textsf{(a-stream)} \qquad \frac{\begin{array}{c} C,\Gamma \vdash {M_1}^\circ \rightsquigarrow C_1 \Rightarrow T_1 \\ C,\Gamma \vdash {M_2}^\circ \rightsquigarrow C_2 \Rightarrow^* \mathsf{stream}^r\, T_2 \\ T_1 \sqcup_{C_2} T_2 \rightsquigarrow T, C_3 \qquad C_4 = C_3 \cup \infty \sqsubseteq \mathrm{SV}(M_1) \end{array}}{C,\Gamma \vdash \mathsf{cons}({M_1}^\circ, {M_2}^\circ) \rightsquigarrow C_4 \Rightarrow \mathsf{stream}^{\widehat{r}}\, T}\ \textsf{(a-cons)}$$

$$\frac{\begin{array}{c} C,\Gamma \vdash M^\circ \rightsquigarrow C_1 \Rightarrow^* \mathsf{stream}^r\, T_1 \qquad \alpha \text{ fresh} \\ (C_1 \cup \widehat{\alpha} \sqsubseteq r), \Gamma(x:T) \vdash P^\circ \rightsquigarrow C_2, P \Rightarrow^* u \qquad C_2, \Gamma(y_1:T_1)(y_2 : \mathsf{stream}^\alpha\, T_1) \vdash N^\circ \Leftarrow P\,[\mathsf{cons}(y_1,y_2)/x] \rightsquigarrow C_3 \end{array}}{C,\Gamma \vdash (\mathsf{case}_{P^\circ}\ x := M^\circ \text{ of } \mathsf{cons}(y_1,y_2) \Rightarrow N^\circ) \rightsquigarrow (C_3 \cup \infty \sqsubseteq \mathrm{SV}(N)) \Rightarrow P\,[M/x]}\ \textsf{(a-case)}$$

$$\frac{\begin{array}{c} T^\star \equiv \Pi \Delta^\star . \mathsf{stream}^\star\, U^\star \qquad C,\Gamma \vdash |T^\star| \rightsquigarrow C_1, \Pi \Delta . \mathsf{stream}^\alpha\, U \Rightarrow^* u \qquad \mathsf{shift}(\Delta,\Delta^\star) = (V^\star, \widehat{\Delta}) \\ C_1, \Gamma(f : \Pi \Delta . \mathsf{stream}^\alpha\, U) \vdash M^\circ \Leftarrow \Pi \widehat{\Delta} . \mathsf{stream}^{\widehat{\alpha}}\, U \rightsquigarrow C_2 \qquad \mathsf{RecCheck}(\alpha, V^\star, C_2 \cup \widehat{\Delta} \sqsubseteq \Delta) = C_3 \end{array}}{C,\Gamma \vdash \mathsf{cofix}\ f{:}T^\star := M^\circ \rightsquigarrow C_3 \Rightarrow \Pi \Delta . \mathsf{stream}^\alpha\, U}\ \textsf{(a-cofix)}$$

Fig. 5. Type-inference algorithm

The next step is implementing this system in the Coq kernel. We also plan to work on the extensions to mutual corecursion and ML-style polymorphism for stages, i.e. $\forall \imath . T(\imath)$.

The model we propose can be easily extended to handle general coinductive types, by interpreting them as the greatest fixed point of monotone operators. The relational interpretation extends naturally: two elements are related in $[\![\mathsf{col}^s]\!]$, for some coinductive type col, if they are related at least until depth $(\!|s|\!)$.

Acknowledgment

The author would like to thank Iliano Cervesato and the anonymous reviewers for their comments and suggestions that helped improve this paper. This work was supported by the Qatar National Research Fund under grant NPRP 09-1107-1-168.

References

[1] The Coq Development Team, *The Coq Reference Manual, version 8.3*, 2009, distributed electronically at http://coq.inria.fr/doc.
[2] U. Norell, "Towards a practical programming language based on dependent type theory," Ph.D. dissertation, Chalmers University of Technology, 2007.
[3] F. Blanqui, "A type-based termination criterion for dependently-typed higher-order rewrite systems," in *RTA*, 2004.
[4] A. Abel, "A polymorphic lambda-calculus with sized higher-order types," Ph.D. dissertation, Ludwig-Maximilians-Universität München, 2006.
[5] J. L. Sacchini, "On type-based termination and dependent pattern matching in the Calculus of Inductive Constructions," Ph.D. dissertation, École Nationale Supérieure des Mines de Paris, 2011.
[6] G. Barthe, B. Grégoire, and F. Pastawski, "CIC$\widehat{}$: Type-based termination of recursive definitions in the Calculus of Inductive Constructions," in *LPAR*, 2006.
[7] B. Grégoire and J. L. Sacchini, "On strong normalization of the calculus of constructions with type-based termination," in *LPAR*, 2010.
[8] T. Altenkirch, "Constructions, inductive types and strong normalization," Ph.D. dissertation, University of Edinburgh, Nov. 1993.
[9] H. Barendregt, "Lambda calculi with types," in *Handbook of Logic in Computer Science*, 1992.
[10] G. Barthe, B. Grégoire, and F. Pastawski, "Practical inference for type-based termination in a polymorphic setting," in *TLCA*, 2005.
[11] E. Giménez, "A calculus of infinite constructions and its application to the verification of communicating systems," Ph.D. dissertation, Ecole Normale Supérieure de Lyon, 1996.
[12] C. McBride, "Let's see how things unfold: Reconciling the infinite with the intensional (extended abstract)," in *CALCO*, 2009.
[13] A. Abel, B. Pientka, D. Thibodeau, and A. Setzer, "Copatterns: Programming infinite structures by observations," 2013, to appear in POPL'13.
[14] P.-A. Melliès and B. Werner, "A generic normalisation proof for pure type systems," in *TYPES*, 1996.
[15] Z. Luo, *Computation and reasoning: a type theory for computer science*. Oxford University Press, Inc., 1994.
[16] B. Werner, "Sets in types, types in sets," in *TACS*, 1997.
[17] T. Coquand, "Infinite objects in type theory," in *TYPES*, 1993.
[18] N. A. Danielsson and T. Altenkirch, "Subtyping, declaratively," in *MPC*, 2010.
[19] A. Abel, "MiniAgda: Integrating sized and dependent types," in *PAR*, 2010.
[20] ——, "Type-based termination, inflationary fixed-points, and mixed inductive-coinductive types," in *FICS*, 2012.
[21] P. Severi and F.-J. de Vries, "Pure type systems with corecursion on streams: from finite to infinitary normalisation," in *ICFP*, 2012.
[22] G. Hutton and M. Jaskelioff, "Representing Contractive Functions on Streams," 2011, submitted to the Journal of Functional Programming.
[23] K. S. Lars Birkedal, Jan Schwinghammer, "A metric model of lambda calculus with guarded recursion," in *FICS*, 2010.
[24] N. R. Krishnaswami and N. Benton, "Ultrametric semantics of reactive programs," in *LICS*, 2011.
[25] P. D. Gianantonio and M. Miculan, "A unifying approach to recursive and co-recursive definitions," in *TYPES*, 2002.

APPENDIX A
SIMPLE METATHEORY OF uCC$\widehat{\omega}$

We prove some metatheoretical results about uCC$\widehat{\omega}$, including substitution, uniqueness of types, and subject reduction (Lemma 2).

Lemma 22 (Confluence)**.** *If $M \approx N$ (resp. $M \approx_{\mathsf{u}} N$), then $M \downarrow N$ (resp. $M \downarrow_{\mathsf{u}} N$).*

Proof: Using the parallel reduction method of Tait and Martin-Löf. □

Definition 23 (Subcontext)**.** *We say that Δ is a subcontext of Γ, denoted $\Delta \leq \Gamma$, if, for all $(x : T) \in \Gamma$, there exists T' such that $(x : T') \in \Delta$ with $T' \leq T$.*

Lemma 24 (Context conversion)**.** *If $\Gamma \vdash M : T$, $\mathsf{WF}(\Delta)$, and $\Delta \leq \Gamma$, then $\Delta \vdash M : T$.*

Proof: By induction on the type derivation. In rules (prod), (abs), (case), and (cofix), we use the following property: if $\Delta \leq \Gamma$, then $\Delta, \Theta \leq \Gamma, \Theta$. □

Lemma 25 (Stage substitution)**.** *If $\Gamma \vdash M : T$ then $\Gamma\,[s/\imath] \vdash M\,[s/\imath] : T\,[s/\imath]$.*

Proof: A straightforward induction fails in rule (cofix), since s might contain the fresh size variable introduced by applying rule (cofix). To solve this issue, we first prove a special case of stage substitution with only size variables:

(*) *if $\Gamma \vdash M : T$ and $\jmath$ is fresh in Γ, M, and T, then $\Gamma\,[\jmath/\imath] \vdash M\,[\jmath/\imath] : T\,[\jmath/\imath]$ with a derivation of the same height.*

We proceed by induction on the type derivation. All cases are easy by applying the IH. In rule (cofix) we use the IH twice if the size variable introduced in the rule is $\jmath$. Then we prove the following result:

(**) *if $\Gamma \vdash M : T$ and s does not contain $\imath$, then $\Gamma\,[s/\imath] \vdash M\,[s/\imath] : T\,[s/\imath]$.*

We proceed by induction on the height h of the type derivation and case analysis on the last rule used. In (cofix) we use (*) if the fresh size variable introduced appears in s.

Using (*) and (**), the main result now follows easily. We consider two cases: if s does not contain $\imath$, the result follows from (**). If s contains $\imath$, we apply (*) and substitute $\imath$ by a fresh size variable κ that does not appear in s. Then we apply (**) substituting κ by s. □

Lemma 26 (Substitution)**.** *If $\Gamma_1(x : T)\Gamma_2 \vdash M : U$, $\Gamma_1 \vdash N : T$, and $\mathrm{SV}(N) = \emptyset$, then $\Gamma_1(\Gamma_2\,[N/x]) \vdash M\,[N/x] : U\,[N/x]$.*

Proof: By induction on the type derivation. □

Lemma 27 (Type validity)**.** *If $\Gamma \vdash M : T$, then $\Gamma \vdash T : u$.*

Proof: By induction on the type derivation, using Lemma 25 in the case of rule (cofix). □

Lemma 28 (Uniqueness of types)**.** *If $\Gamma \vdash M : T_1$ and $\Gamma \vdash M : T_2$, then $|T_1| \approx |T_2|$.*

Proof: By induction on M and the following property: if $T_1 \leq T_2$, then $|T_1| \approx |T_2|$. □

Lemma 29 (Subject reduction)**.** *If $\Gamma \vdash M : T$ and $M \to N$, then $\Gamma \vdash N : T$.*

Proof: By induction on the type derivation and case analysis on the reduction rule. The interesting cases are when reduction occurs at the head. We consider the case rule.

$$\frac{\begin{array}{c}\Gamma \vdash M : \mathsf{stream}^{\widehat{s}}\,T \qquad \Gamma, x{:}\mathsf{stream}^{\widehat{s}}\,T \vdash P : u \\ \Gamma(y_1{:}T)(y_2{:}\mathsf{stream}^{s}\,T) \vdash M' : P\,[\mathsf{cons}(y_1, y_2)/x]] \\ \mathrm{SV}(N) = \emptyset\end{array}}{\Gamma \vdash \left(\begin{array}{l}\mathsf{case}_{|P|}\ x := M\ \mathsf{of} \\ \mathsf{cons}(y_1, y_2) \Rightarrow M'\end{array}\right) : P\,[M/x]}$$

We only consider reduction occurring at the head.

- **$\imath$-redex** We have $M \equiv \mathsf{cons}(M_1, M_2)$ and $N \equiv M'\,[M_1/y_1]\,[M_2/y_2]$. By inversion on the type derivation of M and conversion, we have $\Gamma \vdash M_1 : T$ and $\Gamma \vdash M_2 : \mathsf{stream}^{s}\,T$. The result follows by Lemma 26.
- **ν-redex** We have $M \equiv (\mathsf{cofix}\ f{:}T_1{}^{\star} := M_1)\,\vec{M}$ and

  $$N \equiv \begin{array}{l}\mathsf{case}_{P^{\circ}}\ x := (M_1\,[\mathsf{cofix}\ f{:}T_1{}^{\star} := M_1/f])\,\vec{M}\ \mathsf{of} \\ \mathsf{cons}(y_1, y_2) \Rightarrow M'\end{array}$$

  . By inversion on the type derivation of M, we have $T_1{}^{\star} \equiv \Pi\Delta.\mathsf{stream}^{\imath}\,U$, $\imath\ \mathsf{neg}\ \Delta$, $\Gamma(f : T_1) \vdash M_1 : T_1\,[\widehat{\imath}/\imath]$, and $\Gamma \vdash (\mathsf{cofix}\ f{:}T_1{}^{\star} := M_1) : T_1\,[r/\imath]$. By inversion, $\widehat{s} \sqsubseteq r$, then $r = \widehat{r_1}$ for some r_1. By Lemma 26, $\Gamma \vdash (M_1\,[\mathsf{cofix}\ f{:}T_1{}^{\star} := M_1/f]) : T_1\,[r/\imath]$. Then, $\Gamma \vdash (M_1\,[\mathsf{cofix}\ f{:}T_1{}^{\star} := M_1/f]) : \mathsf{stream}^{s}\,T$. Also by Lemma 26, $\Gamma \vdash P\left[(M_1\,[\mathsf{cofix}\ f{:}T_1{}^{\star} := M_1/f])\,\vec{M}/x\right] : u$. The result follows by conversion (note the use of unrestricted reduction).

□

APPENDIX B
STRONG NORMALIZATION

In this section we give the proofs of the lemmas and theorems stated in Sect. IV. Most of the proofs follow by induction on the relevant structure; we focus on the cases related to streams (see [5] for more details).

Term interpretation: The following lemmas state properties of the term interpretation.

Lemma 30 (Soundness of weakening)**.** *If $[\Gamma\Delta \vdash M]_{\gamma,\delta}$ is defined, and $z \notin \mathrm{FV}(\Delta, M)$, then $[\Gamma(z : T^{\circ})\Delta \vdash M]_{\gamma,\alpha,\delta}$ is defined, and*

$$[\Gamma(z : T^{\circ})\Delta \vdash M]_{\gamma,\alpha,\delta} = [\Gamma\Delta \vdash M]_{\gamma,\delta}\ .$$

Proof: By induction on the structure of M. The case of cofixpoints follows by IH. □

Lemma 31 (Soundness of substitution)**.** *If $\Gamma(x : T)\Delta \vdash M : U$ and $\Gamma \vdash N : T$ are derivable, $\mathrm{SV}(N) = \emptyset$, $\gamma \in [\Gamma]$, $\nu = [\Gamma \vdash N]_{\gamma}$ is defined, $(\gamma, \nu, \delta) \in [\Gamma(x : T)\Delta]$, and $[\Gamma(x : T)\Delta \vdash M]_{\gamma,\nu,\delta}$ is defined, then $[\Gamma, \Delta\,[N/x] \vdash M\,[N/x]]_{\gamma,\delta}$ is defined, and*

$$[\Gamma, \Delta\,[N/x] \vdash M\,[N/x]]_{\gamma,\delta} = [\Gamma(x : T)\Delta \vdash M]_{\gamma,\nu,\delta}\ .$$

Proof: By induction on the structure of M.

- **Cofixpoint** $M \equiv \mathsf{cofix}\ f{:}W^\star := M_1$. Since the interpretation of M is defined, there exists a unique function F in $[\Gamma(x:T)\Delta \vdash W]_{\gamma,\nu,\delta}$ satisfying the fixpoint equations. By IH,

$$[\Gamma\Delta\,[N/x] \vdash W\,[N/x]]_{\gamma,\delta} = [\Gamma(x:T)\Delta \vdash W]_{\gamma,\nu,\delta}$$

$$[\Gamma\Delta\,[N/x]\,(f : |W\,[N/x]\,| \vdash M_1\,[N/x]]_{\gamma,\delta} = [\Gamma(x:T)\Delta(f:|W|) \vdash M_1]_{\gamma,\nu,\delta}$$

Then, the same function F satisfies the cofixpoint equation for $\Gamma\Delta\,[N/x] \vdash \mathsf{cofix}\ f{:}W^\star\,[N/x] := M_1\,[N/x]$, and the result follows.

□

The following lemma states that the interpretation is invariant under reduction. Recall the use of Hilbert's choice operator in the interpretation of cofixpoints: the conditions imposed ensure that ν-reduction is sound.

Lemma 32 (Soundness of subject reduction)**.** *Let $\Gamma \vdash M : T$ and $M \to_{\mathsf{u}} N$. If $[\Gamma \vdash M]_\gamma$ is defined, then $[\Gamma \vdash N]_\gamma$ is defined and*

$$[\Gamma \vdash M]_\gamma = [\Gamma \vdash N]_\gamma$$

Proof: By induction on $M \to N$. The key cases are when M is a redex.

- β**-redex** Follows directly from the definition and Lemma 31.
- ι**-redex** $M \equiv \mathsf{case}_{P^\circ}\ x := \mathsf{cons}(u_1,u_2)\ \mathsf{of}\ \mathsf{cons}(y_1,y_2) \Rightarrow M_1$ and
$$N \equiv M_1\,[u_1/y_1]\,[u_2/y_2]$$
The result follows from Lemma 31.
- ν**-redex**
$$M \equiv \begin{array}{l}\mathsf{case}_{P^\circ}\ x := ((\mathsf{cofix}\ f{:}T^\star := M_1)\,\vec{N_1})\ \mathsf{of}\\ \mathsf{cons}(y_1,y_2) \Rightarrow O\end{array}$$
and
$$N \equiv \begin{array}{l}\mathsf{case}_{P^\circ}\ x := (M_1\,[\mathsf{cofix}\ f{:}T^\star := M_1/f])\,\vec{N_1}\ \mathsf{of}\\ \mathsf{cons}(y_1,y_2) \Rightarrow O\end{array}$$
The result follows from Lemma 31, since by the cofixpoint equation, $[\Gamma \vdash \mathsf{cofix}\ f{:}T^\star := M_1]_\gamma = [\Gamma \vdash M_1\,[\mathsf{cofix}\ f{:}T^\star := M_1/f]]_\gamma$.

The rest of the cases concern the compatible closure of $\to$. They follow easily by direct applications of the IH. □

Relational interpretation: The following lemmas state some properties of the relational interpretation.

Lemma 33 (Stage substitution)**.** *Let s, r be stages, and π a stage assignment. Then $(\!|r\,[s/\imath]|\!)_\pi = (\!|r|\!)_{\pi(\imath:=(\!|s|\!)_\pi)}$.*

Proof: By induction on the structure of r. □

Lemma 34 (Stage monotonicity)**.** *For any stages s, r, such that $s \sqsubseteq r$, and any stage assignment π, $(\!|s|\!)_\pi \le (\!|r|\!)_\pi$.*

Proof: By induction on $s \sqsubseteq r$. Note that, for any s and π, $(\!|s|\!)_\pi \le \omega$. □

The following lemma states that the RI of a simple type does not depend on the derivation.

Lemma 35. *Let T be a term such that $\mathsf{simple}(T)$ and $\mathrm{SV}(T) = \emptyset$. If $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2}$ is defined and $[\Gamma \vdash T]_{\gamma_1} = [\Gamma \vdash T]_{\gamma_2}$, then $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2} = ([\Gamma \vdash T]_{\gamma_1})^2$.*

Proof: By induction on $\mathsf{simple}(T)$. If $T \equiv \mathsf{stream}^s\,T_1$, the result follows from the definition of the RI of streams and the IH on T_1. □

The following lemmas state several soundness properties of the relational interpretation: weakening, stage substitution, substitution, subject reduction and subtyping.

Lemma 36 (Soundness of weakening)**.** *If $[\![\Gamma\Delta \vdash U]\!]^\pi_{(\gamma_1,\delta_1)\sim(\gamma_2,\delta_2)}$ is defined, then $[\![\Gamma(z:T)\Delta \vdash U]\!]^\pi_{(\gamma_1,\alpha_1,\delta_1)\sim(\gamma_2,\alpha_2,\delta_2)}$ is defined and both are equal.*

Proof: If $U \equiv \mathsf{stream}^s\,U_1$, the result follows from Lemma 30. □

Lemma 37 (Soundness of stage substitution)**.** *If $[\![\Gamma \vdash T\,[s/\imath]]\!]^\pi_{\gamma_1\sim\gamma_2}$ is defined, then $[\![\Gamma \vdash T]\!]^{\pi(\imath:=(\!|s|\!)_\pi)}_{\gamma_1\sim\gamma_2}$ is defined, and both are equal.*

Proof: If $T \equiv \mathsf{stream}^r\,T_1$, the result follows from the IH and Lemma 33. □

Lemma 38 (Soundness of substitution)**.** *Let $\Gamma(x:U)\Delta \vdash T : W$ and $\Gamma \vdash N : U$ and $\mathrm{SV}(N) = \emptyset$. Let $\gamma_i \in [\Gamma]$, $\nu_i \equiv [\Gamma \vdash N]_{\gamma_i}$, $(\gamma_i,\nu_i,\delta_i) \in [\Gamma(x:U)\Delta]$, for $i = 1,2$, and $[\![\Gamma(x:U)\Delta \vdash T]\!]^\pi_{\gamma_1,\nu_1,\delta_1\sim\gamma_2,\nu_2,\delta_2}$ are defined. Then*

$$[\![\Gamma\Delta\,[N/x] \vdash T\,[N/x]]\!]^\pi_{(\gamma_1,\delta_1)\sim(\gamma_2,\delta_2)} = [\![\Gamma(x:U)\Delta \vdash T]\!]^\pi_{(\gamma_1,\nu_1,\delta_1)\sim(\gamma_2,\nu_2,\delta_2)}$$

Proof: If $T \equiv \mathsf{stream}^r\,T_1$, the result follows from Lemma 31. □

Lemma 39 (Soundness of subject reduction)**.** *Let $\Gamma \vdash T : u$ and $T \to_{\mathsf{u}} U$, with $\mathsf{simple}(T)$. If $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2}$ is defined, then $[\![\Gamma \vdash U]\!]^\pi_{\gamma_1\sim\gamma_2}$ is defined and*

$$[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2} = [\![\Gamma \vdash U]\!]^\pi_{\gamma_1\sim\gamma_2}$$

Proof: By induction on the structure of T, using Lemma 32. □

The following lemma states that the RI is sound with respect to subtyping.

Lemma 40 (Soundness of subtyping)**.** *Assume $\Gamma \vdash T, U : u$, and $\mathsf{simple}(T,U)$ and $T \le U$. If $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2}$ and $[\![\Gamma \vdash U]\!]^\pi_{\gamma_1\sim\gamma_2}$ are defined, then $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1\sim\gamma_2} \subseteq [\![\Gamma \vdash U]\!]^\pi_{\gamma_1\sim\gamma_2}$*

Proof: We proceed by induction on the derivation of $T \le U$.

- **(st-stream)** We have the derivation
$$\frac{s \sqsubseteq r \qquad T_1 \le U_1}{\mathsf{stream}^r\,T \le \mathsf{stream}^s\,U_1}$$

By IH, $[\![\Gamma \vdash T_1]\!]^{\pi}_{\gamma_1\sim\gamma_2} \subseteq [\![\Gamma \vdash U_1]\!]^{\pi}_{\gamma_1\sim\gamma_2}$. Then, from the definition of the relational interpretation of streams, $[\![\Gamma \vdash \mathsf{stream}^r\, T_1]\!]^{\pi}_{\gamma_1\sim\gamma_2} \subseteq [\![\Gamma \vdash \mathsf{stream}^s\, U_1]\!]^{\pi}_{\gamma_1\sim\gamma_2} \subseteq [\![\Gamma \vdash \mathsf{stream}^s\, U_1]\!]^{\pi}_{\gamma_1\sim\gamma_2}$.

□

The following lemma is used in the proof of soundness of the cofixpoint rule.

Lemma 41 (Soundness of stage monotonicity)**.** *Let s, r be stages such that $s \sqsubseteq r$. If $[\![\Gamma \vdash T]\!]^{\pi}_{\gamma_1\sim\gamma_2}$ is defined, and $\imath$ pos T, then $[\![\Gamma \vdash T]\!]^{\pi(\imath:=s)}_{\gamma_1\sim\gamma_2} \subseteq [\![\Gamma \vdash T]\!]^{\pi(\imath:=r)}_{\gamma_1\sim\gamma_2}$*

Proof: From $\imath$ pos T we know that $T\,[s/\imath] \leq T\,[r/\imath]$. The result follows from the previous lemma. □

Soundness of the Λ-set model: We give the proof of soundness of the Λ-set model. The proof proceeds by induction on the type derivation. The interesting case is the (cofix) rule. The following lemmas show that the use of Hilbert's choice operator is well defined, under some condition guaranteed by the inductive hypotheses in the soundness theorem.

For the following lemmas, let us assume a typing derivation of the form

$$\frac{\begin{array}{cc} T \equiv \Pi\Delta.\mathsf{stream}^{\imath}\, U & \imath \text{ neg } \Delta \\ \imath \notin \mathrm{SV}(\Gamma, U, M) & \Gamma \vdash T : u \\ \Gamma(f : T) \vdash M : T\,[\widehat{\imath}/\imath] & \mathrm{SV}(M) = \emptyset \end{array}}{\Gamma \vdash \mathsf{cofix}\ f{:}|T|^{\imath} := M : T\,[s/\imath]}$$

Furthermore, we assume:

1) $[\Gamma]$ is defined with $\gamma \in [\Gamma]$;
2) $[\Gamma \vdash T]_{\gamma}$ is defined;
3) $[\Gamma(f : T) \vdash M]_{\gamma,\varphi}$ is defined for any φ such that $\gamma, \varphi \in [\Gamma(f : T)]$;
4) $[\![\Gamma]\!]^{\infty}$ is defined;
5) $[\![\Gamma \vdash T]\!]^{\infty(\imath:=\mathfrak{a})}_{\gamma\sim\gamma}$ is defined for any ordinal $\mathfrak{a}$;
6) $[\Gamma(f : T) \vdash M]_{\gamma_1,\varphi_1\sim\gamma_2,\varphi_2} \sqsubset [\![\Gamma \vdash T\,[\widehat{\imath}/\imath]]\!]^{\infty(\imath:=\mathfrak{a})}_{\gamma\sim\gamma}$, for any $\varphi_1 \sim \varphi_2 \sqsubset [\![\Gamma \vdash T]\!]^{\infty(\imath:=\mathfrak{a})}_{\gamma\sim\gamma}$.

Given the above assumptions, we prove that there is a unique function satisfying the cofixpoint equation of Definition. 10. For readability, we omit the context in the interpretations as well as γ.

We give the definition of a function satisfying the cofixpoint equations. Intuitively, the function is constructed by iterating the body of the cofixpoint. Given a set of functions $\Phi \subseteq ([T])_{\circ}$, we write $[M]_{\Phi}$ for the set $\{[M]_{\varphi} : \varphi \in \Phi\}$.

We define a sequence of sets $\{\Phi^{\mathfrak{a}}\}$ for ordinals $\mathfrak{a} \leq \omega$, as follows.

1) $\Phi^0 = \{\varphi : \varphi \sqsubset [T]\}$;
2) $\Phi^{\mathfrak{a}+1} = [M]_{\Phi^{\mathfrak{a}}}$;
3) $\Phi^{\omega} = \bigcap_{\mathfrak{a}<\omega} \Phi^{\mathfrak{a}}$.

The intuition is that the functions in $\Phi^{\mathfrak{a}}$ define the first $\mathfrak{a}$ items of the stream; hence the functions in Φ^{ω} completely defines the stream.

Lemma 42. *For all $\mathfrak{a}$, $\Phi^{\mathfrak{a}} \sim \Phi^{\mathfrak{a}} \sqsubset [\![T]\!]^{\imath:=\mathfrak{a}}$. This means, for all $\varphi_1, \varphi_2 \in \Phi^{\mathfrak{a}}$, $\varphi_1 \sim \varphi_2 \sqsubset [\![T]\!]^{\imath:=\mathfrak{a}}$.*

Proof: By induction on the ordinal $\mathfrak{a}$.

- $\mathfrak{a} = 0$**.** The result follows immediately, as all streams are related.
- $\mathfrak{a} = \mathfrak{b} + 1$**.** The result follows from assumption 6.
- $\mathfrak{a} = \omega$**.** Let $\varphi_1, \varphi_2 \in \Phi^{\omega}$. $(\delta_1, \delta_2) \sqsubset [\![\Delta]\!]^{\imath:=\omega}$. We want to prove that $(\varphi_1(\delta_1), \varphi_2(\delta_2) \sqsubset [\![\mathsf{stream}^{\imath}\, U]\!]^{\imath:=\omega}$. Let $j < \omega$; we prove that $\pi_j(\varphi_1(\delta_1)) = \pi_j(\varphi_2(\delta_2))$. From $\imath$ neg Δ and Lemma 37, $(\delta_1, \delta_2) \sqsubset [\![\Delta]\!]^{\imath:=j+1}$. Since $\varphi_1, \varphi_2 \in \Phi^{j+1}$, by IH, $\varphi_1 \sim \varphi_2 \sqsubset [\![T]\!]^{\imath:=j+1}$. Then, $(\varphi_1(\delta_1), \varphi_2(\delta_2)) \sqsubset [\![\mathsf{stream}^{\imath}\, U]\!]^{\imath:=j+1}$. Then, by definition, $\pi_j(\varphi_1(\delta_1)) = \pi_j(\varphi_2(\delta_2))$.

□

It is easy to see that the sequence $\{\Phi^{\mathfrak{a}}\}_{\mathfrak{a}}$ is decreasing, i.e., if $\mathfrak{a} < \mathfrak{b}$, then $\Phi^{\mathfrak{b}} \subseteq \Phi^{\mathfrak{a}}$. By the previous lemma and Lemma 37, there is at most one function in the set Φ^{ω}. We prove that Φ^{ω} is not empty by constructing an element in it.

Let $\{\varphi^i\}_i$ a sequence of functions $[T]$ defined as follows:

$$\varphi^0 = \delta \sqsubset [\Delta] \mapsto (\emptyset)$$
$$\varphi^{i+1} = [M]_{\varphi^i}$$

Note that $\varphi^i \in \Phi^i$. Let $F \sqsubset [T]$ be defined as follows

- $F(\delta) = (\pi_0(\varphi^1(\delta)), \pi_1(\varphi^2(\delta)), \pi_2(\varphi^3(\delta)), \ldots)$, if $\pi_0(\varphi^i(\delta))$ is defined for all i;
- $F(\delta) = (\pi_0(\varphi^1(\delta)), \pi_1(\varphi^2(\delta)), \ldots, \pi_{n-1}(\varphi^n(\delta)), \emptyset)$, if $\pi_i(\varphi^{i+1}(\delta))$ is defined for $i = 0, \ldots, n-1$ and $\pi_n(\varphi^{n+1}(\delta))$ is not defined

We show that F satisfies the cofixpoint equations. First, we show that F is in Φ^{ω}.

Lemma 43. *For all $\mathfrak{a}$, $F \in \Phi^{\mathfrak{a}}$.*

Proof: By induction on $\mathfrak{a}$.

- $\mathfrak{a} = 0$**.** The result follows immediately since all functions are in Φ^0.
- $\mathfrak{a} = \mathfrak{b} + 1$**.** Let $\varphi \in \Phi^{\mathfrak{a}}$. We want to prove that $(\varphi, F) \sqsubset [\![T]\!]^{\imath:=\mathfrak{a}}$. I.e., for all δ, $j < \mathfrak{a}$, $\pi_j(\varphi(\delta)) \simeq \pi_j(F(\delta))$. We have $\varphi \in \Phi^{\mathfrak{a}} \subseteq \Phi^{\mathfrak{b}}$. By IH, $(\varphi, F) \sqsubset [\![T]\!]^{\imath:=\mathfrak{b}}$; we only need to check the case $j = \mathfrak{b}$, as the case $j < \mathfrak{b}$ is handled by the IH. We have $\pi_{\mathfrak{b}}(F(\delta)) \simeq \pi_{\mathfrak{b}}(\varphi^{\mathfrak{a}}(\delta))$. Since $\varphi^{\mathfrak{a}} \in \Phi^{\mathfrak{a}}$, by Lemma 42, $(\varphi, \varphi^{\mathfrak{a}}) \sqsubset [\![T]\!]^{\imath:=\mathfrak{a}}$. Then, $\pi_{\mathfrak{b}}(\varphi(\delta)) \simeq \pi_{\mathfrak{b}}(\varphi^{\mathfrak{a}}(\delta))$. The result follows.
- $\mathfrak{a} = \omega$**.** The result follows by IH and the definition of Φ^{ω}.

□

Lemma 44. *For all $\mathfrak{a}$, $F \sim [M]_F \sqsubset [\![T]\!]^{\imath:=\mathfrak{a}}$.*

Proof: For any $\mathfrak{a} < \omega$, from the previous lemma, since $F \in \Phi^{\mathfrak{a}}$, then $[M]_F \in \Phi^{\mathfrak{a}+1}$. Then, $[M]_F \in \Phi^{\omega}$. The result follows from Lemma 42. □

Applying the previous lemma to $\mathfrak{a} = \omega$, we get that $F = [M]_F$.

Theorem 45 (Soundness of the Λ-set model)**.** 1) *If $\mathsf{WF}(\Gamma)$, then $[\![\Gamma]\!]^{\pi}$ is defined for any stage valuation π. Furthermore, if $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^{\pi}$, then $(\gamma_1, \gamma_1) \in [\![\Gamma]\!]^{\infty}$.*
2) *If $\Gamma \vdash M : T$, with $\mathrm{SV}(M) = \emptyset$, and $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^{\pi}$, then $[\Gamma \vdash M]_{\gamma_1,\gamma_2}$, $[\![\Gamma \vdash T]\!]^{\pi}_{\gamma_1\sim\gamma_2}$ are defined and*

$$[\Gamma \vdash M]_{\gamma_1,\gamma_2} \sqsubset [\![\Gamma \vdash T]\!]^{\pi}_{\gamma_1\sim\gamma_2}\ .$$

3) *If* $\Gamma \vdash T : \mathsf{Type}_i$, $\mathsf{simple}(T)$, *and* $(\gamma_1, \gamma_2) \in [\![\Gamma]\!]^\pi$, *then* $[\![\Gamma \vdash T]\!]^\pi_{\gamma_1 \sim \gamma_2}$ *is defined, and*

$$[\![\Gamma \vdash T]\!]^\pi_{\gamma_1 \sim \gamma_2} \sqsubset \mathcal{U}_i \ .$$

Proof: By induction on the type derivation. We show only the cases related to streams.

- **(stream)** Follows directly from the definition and the IH.
- **(cons)** M is $\mathsf{cons}(N_1, N_2)$; we have the derivation

$$\frac{\Gamma \vdash N_1 : T \qquad \Gamma \vdash N_2 : \mathsf{stream}^s\, T \qquad \mathrm{SV}(N_2) = \emptyset}{\Gamma \vdash \mathsf{cons}(N_1, N_2) : \mathsf{stream}^{\widehat{s}}\, T}$$

  By IH $[\Gamma \vdash N_1]_{\gamma_1,\gamma_2} \sqsubset [\![\Gamma \vdash T]\!]^\pi_{\gamma_1 \sim \gamma_2}$ and $[\Gamma \vdash N_2]_{\gamma_1,\gamma_2} \sqsubset [\![\Gamma \vdash \mathsf{stream}^s\, T]\!]^\pi_{\gamma_1 \sim \gamma_2}$. We need to show that $([\Gamma \vdash N_1]_{\gamma_1,\gamma_2}, [\Gamma \vdash N_2]_{\gamma_1,\gamma_2}) \sqsubset [\![\Gamma \vdash \mathsf{stream}^{\widehat{s}}\, T]\!]^\pi_{\gamma_1 \sim \gamma_2}$. This follows from the definition of the RI of streams.
- **(case)** M is $\mathsf{case}_{|P|}\ x := N_1\ \mathsf{of}\ \mathsf{cons}(y_1, y_2) \Rightarrow N_2$; we have the derivation

$$\frac{\begin{array}{c}\Gamma \vdash N_1 : \mathsf{stream}^{\widehat{s}}\, T \qquad \Gamma, x{:}\mathsf{stream}^{\widehat{s}}\, T \vdash P : u \\ \Gamma(y_1{:}T)(y_2{:}\mathsf{stream}^s\, T) \vdash N_2 : P\,[\mathsf{cons}(y_1, y_2)/x] \\ \mathrm{SV}(N_2) = \emptyset\end{array}}{\Gamma \vdash \left(\begin{array}{l}\mathsf{case}_{|P|}\ x := N_1\ \mathsf{of} \\ \mathsf{cons}(y_1, y_2) \Rightarrow N_2\end{array}\right) : P\,[N_1/x]}$$

  We have two cases. If $[\Gamma \vdash N_1]_{\gamma_1,\gamma_2} = (\emptyset, \emptyset)$, the result follows immediately. If $[\Gamma \vdash N_1]_{\gamma_1,\gamma_2} = (\langle \alpha_i \rangle_{i=0\ldots}, \langle \alpha'_i \rangle_{i=0\ldots})$, then $[\Gamma \vdash M]_{\gamma_j} = [\Gamma' \vdash N_2]_{\gamma'_j}$ for $j = 1, 2$, where $\Gamma' \equiv \Gamma(y_1{:}T)(y_2{:}\mathsf{stream}\, T)$ and $\gamma'_j = \gamma_j, \alpha_0, \langle \alpha_i \rangle_{i=0\ldots}$. The result follows from the IH on N_2 and soundness of substitution (Lemma 31).
- **(cofix)** M is $\mathsf{cofix}\ f{:}|T|^\imath := N$; we have the derivation

$$\frac{\begin{array}{c}T \equiv \Pi\Delta.\mathsf{stream}^\imath\, U \qquad \imath\ \mathsf{neg}\ \Delta \\ \imath \notin \mathrm{SV}(\Gamma, U, N) \qquad \Gamma \vdash T : u \\ \Gamma(f : T) \vdash N : T\,[\widehat{\imath}/\imath] \qquad \mathrm{SV}(N) = \emptyset\end{array}}{\Gamma \vdash \mathsf{cofix}\ f{:}|T|^\imath := N : T\,[s/\imath]}$$

  We are in conditions of applying Lemmas 42- 44. Then $\varphi_1 = [\Gamma \vdash M]_{\gamma_1}$ and $\varphi_2 = [\Gamma \vdash M]_{\gamma_2}$ are defined and satisfy the cofixpoint equations. We prove that $\varphi_1 \sim \varphi_2 \sqsubset [\![\Gamma \vdash T\,[s/\imath]]\!]^\pi_{\gamma_1 \sim \gamma_2}$. We proceed by induction on $(\!|s|\!)_\pi$, using Lemma 37.
  - $(\!|s|\!)_\pi = 0$**.** The result follows immediately, since all streams are related.
  - $(\!|s|\!)_\pi = \mathfrak{a} + 1$**.** The result follows from the IH on N, Lemma 31 and using the fact that φ_1 and φ_2 satisfy the cofixpoint equations.
  - $(\!|s|\!)_\pi = \omega$**.** We need to show that both streams coincide in position i for any $i < \omega$; this follows from the IH on $(\!|s|\!)_\pi$.

□

Theorem 15 follows from the above, since it is a special case.

Finally, we prove that a term realizes its interpretation (Lemma 17).

Proof of Lemma 17: We proceed by induction on the derivation of $\Gamma \vdash M : T$. We only consider the cases related to streams.

- **(stream)** M is $\mathsf{stream}^s\, T$. We need to show that $M\theta \in \mathrm{SN}$. By IH on T, $T\theta \in \mathrm{SN}$. The result follows.
- **(cons)** M is $\mathsf{cons}(N_1, N_2)$; we have the derivation

$$\frac{\Gamma \vdash N_1 : T \qquad \Gamma \vdash N_2 : \mathsf{stream}^s\, T \qquad \mathrm{SV}(N_1) = \emptyset}{\Gamma \vdash \mathsf{cons}(N_1, N_2) : \mathsf{stream}^{\widehat{s}}\, T}$$

  By IH, $N_1\theta \models_{[\![\Gamma \vdash T_1]\!](\gamma_1 \sim \gamma_2, \pi)} [\Gamma \vdash N_1]_{\gamma_1,\gamma_2}$ and $N_2\theta \models_{[\![\Gamma \vdash \mathsf{stream}^s\, T_1]\!](\gamma_1 \sim \gamma_2, \pi)} [\Gamma \vdash N_2]_{\gamma_1,\gamma_2}$. Note that $\downarrow \mathsf{cons}(N_1\theta, N_2\theta) \to_{\mathrm{wh}} \mathsf{cons}(N_1\theta, N_2\theta)$. The result follows from the relational interpretation of streams.
- **(case)** M is $\mathsf{case}_{|P|}\ x := N_1\ \mathsf{of}\ \mathsf{cons}(y_1, y_2) \Rightarrow N_2$; we have the derivation

$$\frac{\begin{array}{c}\Gamma \vdash N_1 : \mathsf{stream}^{\widehat{s}}\, T \qquad \Gamma, x{:}\mathsf{stream}^{\widehat{s}}\, T \vdash P : u \\ \Gamma(y_1{:}T)(y_2{:}\mathsf{stream}^s\, T) \vdash N_2 : P\,[\mathsf{cons}(y_1, y_2)/x] \\ \mathrm{SV}(N_2) = \emptyset\end{array}}{\Gamma \vdash \left(\begin{array}{l}\mathsf{case}_{|P|}\ x := N_1\ \mathsf{of} \\ \mathsf{cons}(y_1, y_2) \Rightarrow N_2\end{array}\right) : P\,[N_1/x]}$$

  If $[\Gamma \vdash N_1]_{\gamma_1,\gamma_2} = (\emptyset, \emptyset)$, by IH $\downarrow N_1\theta \to^*_{\mathrm{wh}} M' \in \mathrm{AT}$. Then $M\theta \to^*_{\mathrm{wh}} M'' \in \mathrm{AT}$ for some M''; by IH, $|P|\theta$ and $N_1\theta$ are SN, and the result follows. If $[\Gamma \vdash N_1]_{\gamma_1,\gamma_2} = (\langle \alpha_i \rangle_{i=0\ldots}, \langle \alpha'_i \rangle_{i=0\ldots})$, then by IH, $\downarrow N_1\theta \to^*_{\mathrm{wh}} \mathsf{cons}(N'_1, N'_2)$. Then $M\theta \to^*_{\mathrm{wh}} N_2\theta\,[N'_1/y_1]\,[N'_2/y_2]$; the result follows from the IH on N_2, and the fact that realizers are closed under weak-head expansion.
- **(cofix)** M is $\mathsf{cofix}\ f{:}|T|^\imath := M_1$; we have the derivation

$$\frac{\begin{array}{c}T \equiv \Pi\Delta.\mathsf{stream}^\imath\, U \qquad \imath\ \mathsf{neg}\ \Delta \\ \imath \notin \mathrm{SV}(\Gamma, U, M_1) \qquad \Gamma \vdash T : u \\ \Gamma(f : T) \vdash M_1 : T\,[\widehat{\imath}/\imath] \qquad \mathrm{SV}(M_1) = \emptyset\end{array}}{\Gamma \vdash \mathsf{cofix}\ f{:}|T|^\imath := M_1 : T\,[s/\imath]}$$

  Let (φ_1, φ_2) be $[\Gamma \vdash M]_{\gamma_1,\gamma_2}$. By Theorem 15, $\varphi_i = [\Gamma(f : T) \vdash M_1]_{(\gamma_i, \varphi_i)}$, for $i = 1, 2$. We need to show that for all $\vec{N}$ and $(\delta_1, \delta_2) \sqsubset [\![\Gamma \vdash \Delta\,[s/\imath]]\!]^\pi_{\gamma_1 \sim \gamma_2}$ such that $\vec{N} \models_{[\![\Gamma \vdash \Delta[s/\imath]]\!](\gamma_1 \sim \gamma_2, \pi)} (\delta_1, \delta_2)$,

$$(\mathsf{cofix}\ f{:}T^\star := M_1)\theta\,\vec{N} \models_{[\![\Gamma, \Delta \vdash \mathsf{stream}^s\, U]\!](\gamma_1, \delta_1 \sim \gamma_2, \delta_2, \pi)} (\varphi_1(\delta_1), \varphi_2(\delta_2))$$

  We proceed by induction on $(\!|s|\!)_\pi$.
  - If $(\!|s|\!)_\pi = 0$, we need to prove that $(\mathsf{cofix}\ f{:}T^\star := M_1)\theta\,\vec{N} \in \mathrm{SN}$. By the IH, $M_1\theta, T^\star\theta \in \mathrm{SN}$. By assumption, $\vec{N} \in \mathrm{SN}$. The result follows.
  - Assume that $(\!|s|\!)_\pi = \mathfrak{a} + 1$. By IH,

    $$(\mathsf{cofix}\ f{:}T^\star := M_1)\theta \models_{[\![\Gamma \vdash T]\!](\gamma_1 \sim \gamma_2, \pi(\imath := \mathfrak{a}))} (\varphi_1, \varphi_2)$$

    Let $\theta' = \theta(f \mapsto (\mathsf{cofix}\ f{:}T^\star := M_1)\theta)$. Then, $\theta' \models^{\Gamma(f:T)}_{\pi(\imath := \mathfrak{a})} (\gamma_1, \varphi_1), (\gamma_2, \varphi_2)$. By the outer IH on

M_1,

$$M_1\theta' \models_{[\![\Gamma \vdash T[\widehat{\imath}/\imath]]\!](\gamma_1\sim\gamma_2,\pi(\imath:=\mathfrak{a}))} [\Gamma(f:T) \vdash M_1]_{\gamma_1,\varphi_1\sim\gamma_2,\varphi_2}$$

Note that $[\![\Gamma \vdash T[\widehat{\imath}/\imath]]\!]^{\pi(\imath:=\mathfrak{a})}_{\gamma_1\sim\gamma_2} = [\![\Gamma \vdash T]\!]^{\pi(\imath:=\mathfrak{a}+1)}_{\gamma_1\sim\gamma_2}$ by Lemma 37. Let $\vec{N}$ and (δ_1,δ_2) such that $(\delta_1,\delta_2) \sqsubset [\![\Gamma \vdash \Delta]\!]^{\pi(\imath:=\mathfrak{a}+1)}_{\gamma_1\sim\gamma_2}$ and $\vec{N} \models_{[\![\Gamma\vdash\Delta]\!](\gamma_1\sim\gamma_2,\pi(\imath:=\mathfrak{a}+1))} (\delta_1,\delta_2)$. Then,

$$M_1\theta'\,\vec{N} \models_{[\![\Gamma\Delta\vdash \mathsf{stream}^\imath\, U]\!](\gamma_1,\delta_1\sim\gamma_2,\delta_2,\pi(\imath:=\mathfrak{a}+1))} (\varphi_1(\delta_1),\varphi_2(\delta_2))$$

If $(\varphi_1(\delta_1),\varphi_2(\delta_2)) = (\emptyset,\emptyset)$ we proceed as in the previous case. Otherwise, $\downarrow(M_1\theta'\,\vec{N}) \to^*_{\mathrm{wh}} \mathsf{cons}(N_1,N_2)$ for some N_1 and N_2. Then, from

$$\downarrow((\mathsf{cofix}\ f{:}T^\star := M_1)\theta\,\vec{N}) \to_{\mathrm{wh}} \downarrow(M_1\theta'\,\vec{N})$$

the result follows.

– Finally, we consider the case $(\!|s|\!)_\pi = \omega$. Let $\vec{N}$ and $(\delta_1,\delta_2) \sqsubset [\![\Gamma \vdash \Delta[\infty/\imath]]\!]^{\pi}_{\gamma_1\sim\gamma_2}$ such that $\vec{N} \models_{[\![\Gamma\vdash\Delta[\infty/\imath]]\!](\gamma_1\sim\gamma_2,\pi)} (\delta_1,\delta_2)$. We want to show that

$$M_1\theta'\,\vec{N} \models_{[\![\Gamma\Delta\vdash \mathsf{stream}^\imath\, U]\!](\gamma_1,\delta_1\sim\gamma_2,\delta_2,\pi(\imath:=\omega))} (\varphi_1(\delta_1),\varphi_2(\delta_2))$$

Recall that $[\![\Gamma\Delta \vdash \mathsf{stream}^\imath\, U]\!]^{\pi(\imath:=\omega)}_{\gamma_1,\delta_1\sim\gamma_2,\delta_2} = \bigcap_{\mathfrak{a}<\omega} [\![\Gamma\Delta \vdash \mathsf{stream}^\imath\, U]\!]^{\pi(\imath:=\mathfrak{a})}_{\gamma_1,\delta_1\sim\gamma_2,\delta_2}$. Since $\imath$ neg Δ, $\vec{N} \models_{[\![\Gamma\vdash\Delta]\!](\gamma_1\sim\gamma_2,\pi(\imath:=\mathfrak{a}))} (\delta_1,\delta_2)$ for any $\mathfrak{a}$. The result follows by the main IH.

□

Appendix C
Type Inference

We give the proofs of soundness and completeness of the type inference algorithm.

Before stating soundness and completeness of the type inference algorithm, we need to prove some simple properties about the algorithm.

Lemma 46. *Assume that $\rho_1 \le_{V_1} \rho_2$. If V_1 pos T, then $\rho_1 T \le \rho_2 T$. If V_1 neg T, then $\rho_2 T \le \rho_1 T$.*

Proof: By induction on the definition of positivity. □

Lemma 47. *If $\rho \models (\infty \sqsubseteq \mathrm{SV}(M))$, then $\mathrm{SV}(\rho M) = \emptyset$.*

Proof: We must have $\rho(\alpha) = \infty$ for all $\alpha \in \mathrm{SV}(M)$. The result follows by induction on terms. □

Lemma 48.
- *$T \le T[\widehat{\imath}/\imath]$ if and only if $\imath$ pos T.*
- *$T[\widehat{\imath}/\imath] \le T$ if and only if $\imath$ neg T.*

Proof: By induction on the relevant derivation. □

Lemma 49. *If $T_1 \sqcup_C T_2 \rightsquigarrow T', C'$ and $\rho \models C'$, then $\rho T_1, \rho T_2 \le \rho T$*

Lemma 50. *If $\rho \models C$, $\rho T_1, \rho T_2 \le T$, then there exists $T_1 \sqcup_C T_2 \rightsquigarrow T', C'$ and $\rho' \models C'$ such that $\rho' T' \le T$.*

Lemma 51 (Soundness of type inference)**.**

1) *If $C, \Gamma \vdash M^\circ \Leftarrow T \rightsquigarrow C', M$, then for all $\rho \models C'$, $\rho\Gamma \vdash \rho M : \rho T$.*
2) *If $C, \Gamma \vdash M^\circ \rightsquigarrow C', M \Rightarrow T$, then for all $\rho \models C'$, $\rho\Gamma \vdash \rho M : \rho T$.*

Proof: We proceed by simultaneous induction on the respective derivation. We only consider the cases related to streams.

- **(a-stream)** M° is $\mathsf{stream}\, T_1$. Take ρ such that $\rho \models C_1$. By IH, $\rho\Gamma \vdash \rho T_1 : \mathsf{Type}_0$. The result follows by applying rule (stream).
- **(a-cons)** M° is $\mathsf{cons}({M_1}^\circ, {M_2}^\circ)$. Take ρ such that $\rho \models C_3 \cup \infty \sqsubseteq \mathrm{SV}(M_1)$. By IH, $\rho\Gamma \vdash \rho M_1 : \rho T_1$ and $\rho\Gamma \vdash \rho M_2 : \rho(\mathsf{stream}^r\, T_2)$. By Lemma 47, $\mathrm{SV}(\rho M_1) = \emptyset$. The result follows from Lemma 49, two applications of conversion, and rule (cons).
- **(a-case)** M° is $\mathsf{case}_{P^\circ}\ x := {M_1}^\circ$ of $\mathsf{cons}(y_1,y_2) \Rightarrow {M_2}^\circ$. Take ρ such that $\rho \models C_3 \cup \infty \sqsubseteq \mathrm{SV}(M_2)$. By IH $\rho\Gamma \vdash \rho M_1 : \rho(\mathsf{stream}^r\, T_1)$. Since $\rho\widehat{\alpha} \sqsubseteq \rho r$, using conversion, we have $\rho\Gamma \vdash \rho M_1 : \rho(\mathsf{stream}^{\widehat{\alpha}}\, T_1)$. By IH—and conversion—$\rho\Gamma(x : \rho(\mathsf{stream}^{\widehat{\alpha}}\, T_1)) \vdash \rho P : u$. Again, by IH, $\rho\Gamma(y_1 : \rho T_1)(y_2 : \rho(\mathsf{stream}^\alpha\, T_1)) \vdash \rho M_2 : \rho(P\,[\mathsf{cons}(y_1,y_2)/x])$. By Lemma 47, $\mathrm{SV}(\rho M_2) = \emptyset$. The result follows by applying rule (case).
- **(a-cofix)** M is $\mathsf{cofix}\ f{:}T^\star := {M_1}^\circ$. Take ρ such that $\rho \models C_3$. By (SRC) there exists ρ' and $\imath$ such that $\rho'(\alpha) = \imath$ and $\rho' \models C_2 \cup \widehat{\Delta} \sqsubseteq \Delta$, $[\rho(\alpha)/\imath]\,\rho' \le_{V^\star} \rho$, and $\lfloor \rho'(V^\star) \rfloor = \imath$, and $\rho(\alpha_0) \ne \imath$ for all $\alpha_0 \notin V^\star$.
  Since $\mathrm{SV}(\Gamma, U, M_1) \cap V^\star = \emptyset$, we have $\rho\Gamma = \rho'\Gamma$, $\rho U = \rho' U$ and $\rho M_1 = \rho' M_1$. By IH, using ρ',

$$\rho\Gamma \vdash \rho'(\Pi\Delta.\mathsf{stream}^\alpha\, U) : u$$
$$\rho\Gamma(f : \rho'(\Pi\Delta.\mathsf{stream}^\alpha\, U)) \vdash \rho M_1 : \rho'(\Pi\widehat{\Delta}.\mathsf{stream}^{\widehat{\alpha}}\, U)$$

  Since $\rho'(V^\star) = \imath$ and $\mathsf{shift}(\Delta,\Delta^\star) = (V^\star, \widehat{\Delta})$, we have $\rho'\widehat{\Delta} \equiv ([\widehat{\imath}/\imath]\,\rho')\Delta$. From $\rho' \models \widehat{\Delta} \sqsubseteq \Delta$ we have $([\widehat{\imath}/\imath]\,\rho')\Delta \le \rho'\Delta$. By Lemma 48, $\imath$ neg Δ. Note that $\rho'(\Pi\widehat{\Delta}.\mathsf{stream}^{\widehat{\alpha}}\, U) = ([\widehat{\imath}/\imath]\,\rho')(\Pi\Delta.\mathsf{stream}^\alpha\, U) = (\rho'(\Pi\Delta.\mathsf{stream}^\alpha\, U))[\imath := \widehat{\imath}]$. By rule (cofix) using $s = \rho(\alpha)$, we have $\rho\Gamma \vdash \mathsf{cofix}\ f{:}T^\star := \rho M_1 : \rho'(\Pi\Delta.\mathsf{stream}^\alpha\, U)[\imath := s]$. By Lemma 46, $\rho'(\Pi\Delta.\mathsf{stream}^\alpha\, U)[\imath := s] \le \rho(\Pi\Delta.\mathsf{stream}^\alpha\, U)$. The result follows by conversion.

□

Lemma 52 (Completeness of type inference)**.**

1) *If $\rho\Gamma \vdash M : \rho T$, $\rho \models C$, and $\mathrm{SV}(\Gamma, T) \subseteq V$, then there exist C', M', ρ' such that $\rho' \models C'$, $\rho =_V \rho'$, $\rho' M' = M$, and $C, \Gamma \vdash |M| \Leftarrow T \rightsquigarrow C', M'$*
2) *If $\rho\Gamma \vdash M : T$, $\rho \models C$, and $\mathrm{SV}(\Gamma) \subseteq V$, there exist C', M', T', ρ' such that $\rho' \models C'$, $\rho' T' \le T$, $\rho' =_V \rho$, $\rho' M' = M$, and $C, \Gamma \vdash |M| \rightsquigarrow C', M' \Rightarrow T'$.*

Proof: We first prove that (2) implies (1). Let $\rho\Gamma \vdash M : \rho T$, and take C and V such that $\rho \models C$ and $\mathrm{SV}(\Gamma, T) \subseteq V$. By (2), there exists C_1, M_1, T_1, ρ_1 such that $\rho_1 \models C_1$, $\rho_1 T_1 \leq \rho T$, $\rho_1 =_V \rho$, $\rho_1 M_1 = M$, and $C, \Gamma \vdash |M| \rightsquigarrow C_1, M_1 \Rightarrow T_1$. Since $\mathrm{SV}(T) \subseteq V$, $\rho T = \rho_1 T$. The result follows.

We prove (2) by induction on the type derivation.

- **(stream)** M is $\mathsf{stream}^s\, T_1$; we have the derivation

$$\frac{\rho\Gamma \vdash T_1 : \mathsf{Type}_0}{\rho\Gamma \vdash \mathsf{stream}^s\, T_1 : \mathsf{Type}_0}$$

  By (1), there exists C', T_1', and ρ' such that $\rho' \models C'$, $\rho' =_V \rho$, $\rho' T_1' = T_1$, and $C, \Gamma \vdash |T_1| \Leftarrow \mathsf{Type}_0 \rightsquigarrow C', T_1'$. Take a fresh α such that $\alpha \notin V$. Then $C' \cup \alpha \sqsubseteq \infty$, $\mathsf{stream}^\alpha\, T_1'$, Type_0, and $[s/\alpha]\,\rho'$ satisfy the requirements.
- **(cons)** M is $\mathsf{cons}(M_1, M_2)$; we have the derivation

$$\frac{\begin{array}{c}\rho\Gamma \vdash M_1 : T \\ \rho\Gamma \vdash M_2 : \mathsf{stream}^s\, T \qquad \mathrm{SV}(M_1) = \emptyset\end{array}}{\rho\Gamma \vdash \mathsf{cons}(M_1, M_2) : \mathsf{stream}^{\widehat{s}}\, T}$$

  By IH, there exists C_1, M_1', T_1 and ρ_1 such that $\rho_1 \models C_1$, $\rho_1 T_1 \leq T$, $\rho_1 =_V \rho$, $\rho_1 M_1' = M_1$, and $C, \Gamma \vdash |M_1| \rightsquigarrow C_1, M_1' \Rightarrow T_1$. By IH, using $V' \supseteq \mathrm{SV}(T_1)$, there exists C_2, M_2', T_2 and ρ_2 such that $\rho_2 \models C_2$, $\rho_2 T_2 \leq T$, $\rho_2 =_{V'} \rho_1$, $\rho_2 M_2' = M_1$, and $C, \Gamma \vdash |M_2| \rightsquigarrow C_2, M_2' \Rightarrow T_2$. Since $\rho_2 T_1, \rho_2 T_2 \leq T$, the result follows from Lemma 50.
- **(case)** M is $\mathsf{case}_{|P|}\ x := M_1\ \mathsf{of}\ \mathsf{cons}(y_1, y_2) \Rightarrow M_2$

$$\frac{\begin{array}{c}\rho\Gamma \vdash M_1 : \mathsf{stream}^{\widehat{s}}\, T \qquad \rho\Gamma, x{:}\mathsf{stream}^{\widehat{s}}\, T \vdash P : u \\ \rho\Gamma(y_1 : T)(y_2 : \mathsf{stream}^s\, T) \vdash M_2 : P\,[\mathsf{cons}(y_1, y_2)/x] \\ \mathrm{SV}(M_2) = \emptyset\end{array}}{\rho\Gamma \vdash \left(\begin{array}{l}\mathsf{case}_{|P|}\ x := M_1\ \mathsf{of} \\ \mathsf{cons}(y_1, y_2) \Rightarrow M_2\end{array}\right) : P\,[M_1/x]}$$

  By IH, there exists C_1, M_1', T_1 and ρ_1 such that $\rho_1 \models C_1$, $\rho_1 T_1 \leq \mathsf{stream}^{\widehat{s}}\, T$, $\rho_1 =_V \rho$, $\rho_1 M_1' = M_1$, and $C, \Gamma \vdash |M_1| \rightsquigarrow C_1, M_1' \Rightarrow T_1$. Then $\mathsf{whnf}(T_1) = \mathsf{stream}^r\, T_1'$ with $\widehat{s} \sqsubseteq \rho(r)$. Take a fresh α, and set $\rho_1(\alpha) = s$. The result follows by applying the IH to the derivations of P and M_2.
- **(cofix)** M is $\mathsf{cofix}\ f{:}|T|^\iota := M_1$; we have the derivation

$$\frac{\begin{array}{c}T \equiv \Pi\Delta.\mathsf{stream}^\iota\, U \qquad \iota\ \mathsf{neg}\ \Delta \\ \iota \notin \mathrm{SV}(\Gamma, U, M) \qquad \rho\Gamma \vdash T : u \\ \rho\Gamma(f : T) \vdash M_1 : T\,[\widehat{\iota}/\iota] \qquad \mathrm{SV}(M) = \emptyset\end{array}}{\rho\Gamma \vdash (\mathsf{cofix}\ f{:}|T|^\iota := M_1) : T\,[s/\iota]}$$

  By IH on the derivation of T, there exists C_T, T', W_T, ρ_T such that $\rho_T \models C_T$, $\rho_T W_T \leq u$, $\rho_T =_V \rho$, $\rho_T T' = T$, and $C, \Gamma \vdash |T| \rightsquigarrow C_T, T' \Rightarrow W$.

  Let $T' = \Pi\Delta'.\mathsf{stream}^\alpha\, U'$ and $|T|^\iota = \Pi|\Delta|^\iota.\mathsf{stream}^\alpha\, U^\circ$. Let $(V^\star, \widehat{\Delta}) = \mathsf{shift}(\Delta', |\Delta|^\iota)$. Then, from $\rho_T T' = T$, we have $\lfloor \rho_T(V^\star) \rfloor = \iota$ and, in particular, $\rho_T(\alpha) = \iota$.

  Note that $T\,[\widehat{\iota}/\iota] = \rho_T T'\,[\widehat{\iota}/\iota] = \rho_T(\Pi\widehat{\Delta}.\mathsf{stream}^{\widehat{\alpha}}\, U)$. By IH on the derivation of M_1, using $V' \supseteq V \cup V^\star$, there exist C_B, M_B, ρ_B such that $\rho_B \models C_B$, $\rho_B =_{V'} \rho_T$, $\rho_B M_B = M_1$, and $C_T, \Gamma \vdash M_1 \Leftarrow \Pi\widehat{\Delta}.\mathsf{stream}^{\widehat{\alpha}}\, U \rightsquigarrow C_B, M_B$. Since $V^\star \subseteq V'$, we have $\lfloor \rho_B(V^\star) \rfloor = \iota$. Since ι does not occur in $\rho\Gamma = \rho_B\Gamma$, $M_1 = \rho_B M_B$, and $\rho_B U' = \rho_T U' = U$, then $\lfloor \rho_B(\mathrm{SV}(C_B)) \setminus V^\star) \rfloor \neq \iota$. Since ι neg Δ, $\widehat{\Delta} \leq \Delta$. Then $\rho_B\widehat{\Delta} \leq \rho_B\Delta$ and $\rho_B \models \widehat{\Delta} \sqsubseteq \Delta$. We have then $\rho_B \models C_B \cup \widehat{\Delta} \sqsubseteq \Delta$. By (CRC), $C_f = \mathsf{RecCheck}(\alpha, V^\star, C_B \cup \widehat{\Delta} \sqsubseteq \Delta)$ is defined and $\rho_B \models C_f$. Then $C_f, \mathsf{cofix}\ f{:}|T|^\iota := M_B, T', [s/\iota]\,\rho_B$ satisfy the required conditions.

□