

AN OPERATIONAL INTERPRETATION OF COINDUCTIVE TYPES

LUKASZ CZAJKA

TU Dortmund University, Dortmund, Germany
e-mail address: lukaszcz@mimuw.edu.pl

ABSTRACT. We introduce an operational rewriting-based semantics for strictly positive nested higher-order (co)inductive types. The semantics takes into account the “limits” of infinite reduction sequences. This may be seen as a refinement and generalization of the notion of productivity in term rewriting to a setting with higher-order functions and with data specified by nested higher-order inductive and coinductive definitions. Intuitively, we interpret lazy data structures in a higher-order functional language by potentially infinite terms corresponding to their complete unfoldings.

We prove an approximation theorem which essentially states that if a term reduces to an arbitrarily large finite approximation of an infinite object in the interpretation of a coinductive type, then it infinitarily (i.e. in the “limit”) reduces to an infinite object in the interpretation of this type. We introduce a sufficient syntactic correctness criterion, in the form of a type system, for finite terms decorated with type information. Using the approximation theorem, we show that each well-typed term has a well-defined interpretation in our semantics.

1. INTRODUCTION

It is natural to consider an interpretation of coinductive types where the elements of a coinductive type ν are possibly infinite terms. Each finite term of type ν containing fixpoint operators then “unfolds” to a possibly infinite term without fixpoint operators in the interpretation of ν . For instance, one would interpret the type of binary streams as the set of infinite terms of the form $b_1 :: b_2 :: \dots$ where $b_1 \in \{0, 1\}$ and $::$ is an infix notation for the stream constructor. Then any fixpoint definition of a term of this type should “unfold” to such an infinite term. This kind of interpretation corresponds closely to a naive understanding of infinite objects and coinductive types.

This paper is devoted to a study of such an interpretation in the context of infinitary rewriting. Infinitary rewriting extends term rewriting by infinite terms and transfinite reductions. This enables the consideration of “limits” of terms under infinite reduction sequences.

Key words and phrases: coinductive types, productivity, infinitary rewriting, programming language semantics, functional programming.

Supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement number 704111.

We consider a combination of simple function types with strictly positive nested higher-order inductive and coinductive types. An example of a higher-order coinductive type is the type of trees with potentially infinite branches and two kinds of nodes: nodes with a list of finitely many children and nodes with infinitely many children specified by a function on natural numbers. In our notation this type may be represented as the coinductive definition $\text{Tree}_2 = \text{CoInd}\{c_1 : \text{List}(\text{Tree}_2) \rightarrow \text{Tree}_2, c_2 : (\text{Nat} \rightarrow \text{Tree}_2) \rightarrow \text{Tree}_2\}$ which intuitively specifies that each element of Tree_2 is a possibly infinite term which has one of the forms:

- $c_1(t_1 :: t_2 :: \dots :: t_n :: \text{nil})$ where each t_i is an element of Tree_2 and $::$ is the finite list constructor, or
- $c_2 f$ where f is a term which represents a function from Nat to Tree_2 .

We interpret each type τ as a subset $\llbracket \tau \rrbracket$ of the set \mathbb{T}^∞ of finite and infinite terms. This interpretation may be seen as a refinement and generalization of the notion of productivity in term rewriting to a setting with higher-order functions and more complex (co)inductive data structures. From a programming language perspective, we essentially interpret lazy data structures in a higher-order functional language by potentially infinite terms corresponding to their complete unfoldings (i.e. their “limits” under infinite reductions).

For example, the interpretation $\llbracket \text{Strm} \rrbracket$ of the coinductive type Strm of streams of natural numbers with a single constructor $\text{cons} : \text{Nat} \rightarrow \text{Strm} \rightarrow \text{Strm}$ consists of all infinite terms of the form $\text{cons } n_0(\text{cons } n_1(\dots))$ where $n_k \in \llbracket \text{Nat} \rrbracket$ for $k \in \mathbb{N}$. The interpretation $\llbracket \text{Strm} \rightarrow \text{Strm} \rrbracket$ of an arrow type $\text{Strm} \rightarrow \text{Strm}$ is the set of all terms t such that for every $u \in \llbracket \text{Strm} \rrbracket$ there is $u' \in \llbracket \text{Strm} \rrbracket$ with $tu \rightarrow^\infty u'$, where \rightarrow^∞ denotes the infinitary reduction relation (so u' is the “limit” of a reduction starting with tu). This means that t is productive – it computes (in the limit) a stream when given a stream as an argument, producing any initial finite segment of the result using only an initial finite segment of the argument. Note that the argument u is just any infinite stream of natural numbers – it need not even be computable. This corresponds with the view that arguments to a function may come from an outside “environment” about which nothing is assumed, e.g., the argument may be a stream of requests for an interactive program.

One could informally argue that including infinite objects explicitly is not necessary, because it suffices to consider finite “approximations” u_n of “size” n of an infinite argument object u (which itself is possibly not computable), and if tu_n reduces to progressively larger approximations of an infinite object for progressively larger n , then this “defines” the application of t to u , because to compute any finite part of the result it suffices to take a sufficiently large approximation as an argument. We actually make this intuition precise in the framework of infinitary rewriting. We show that if for every approximation u_n of size n of an infinite object u the application tu_n reduces to an approximation of an infinite object of the right type, with the result approximations getting larger as n gets larger, then there is a reduction starting from tu which “in the limit” produces an infinite object of the right type. For nested higher-order (co)inductive types this result turns out to be non-trivial.

The result mentioned above actually follows from the approximation theorem which is the central technical result of this paper. It may be stated as follows: if $t \rightarrow^\infty t_n \in \llbracket \nu \rrbracket^n$ for each $n \in \mathbb{N}$ then there is t' with $t \rightarrow^\infty t' \in \llbracket \nu \rrbracket$, where ν is a coinductive type and $\llbracket \nu \rrbracket^n$ is the set of approximations of size n of the (typically infinite) objects of type ν (i.e. of the terms in $\llbracket \nu \rrbracket$).

In the second part of the paper we consider *finite* terms decorated with type annotations. We present a type system which gives a sufficient syntactic correctness criterion for such

terms. The system enables reasoning about sizes of (co)inductive types, similarly as in systems with sized types. Using the approximation theorem we show soundness: if a finite decorated term t may be assigned type τ in our type system, then there is $t' \in \llbracket \tau \rrbracket$ such that $|t| \rightarrow^\infty t'$, where $|t|$ denotes the term t with type decorations erased. This means that every typable term t has a well-defined interpretation in the corresponding type, which may be obtained as a limit of a reduction sequence starting from $|t|$.

Our definition of the rewriting semantics is natural and relatively straightforward. It is not difficult to prove it sound for a restricted form of non-nested first-order (co)inductive types. However, once we allow parameterized nested higher-order inductive and coinductive types significant complications occur because of the alternation of least and greatest fixpoints in the definitions. Our main technical contribution is the proof of the approximation theorem. This proof involves some heavy infinitary rewriting machinery, but just to apply the theorem no deep familiarity with infinitary rewriting is needed.

The main purpose of this paper is to define an infinitary rewriting semantics, to precisely state and prove the approximation theorem, and to show that the approximation theorem may be used to derive soundness of the rewriting semantics for systems based on sized types. The type system itself presented in the second part of the paper is not a significant improvement over the state-of-the-art in type systems based on sized types. It is mostly intended as an illustration of a system for which our rewriting semantics is particularly perspicuous.

1.1. Related work. The notion of productivity dates back to the work of Dijkstra [14], and the later work of Sijtsma [44]. Our rewriting semantics may be considered a generalization of Ishihara's definition of productivity in algorithmic systems [25], of Zantema's and Raffelsieper's definition of productivity in infinite data structures [46], and of the definition of stream productivity [16, 15, 19]. In comparison to our setting, the infinite data structures considered before in term rewriting literature are very simple. None of the papers mentioned allow higher-order functions or higher-order (co)inductive types. The relative difficulty of our main results stems from the fact that the data structures we consider may be much more complex.

Infinitary rewriting was introduced in [29, 28, 30]. See [27] for more references and a general introduction.

In the context of type theory, infinite objects were studied by Martin-Löf [38] and Coquand [9]. Gimenez [21] introduced the guardedness condition to incorporate coinductive types and corecursion into dependent type theory, which is the approach currently used in Coq. Sized types are a long-studied approach for ensuring termination and productivity in type theories [24, 7, 2, 4]. In comparison to previous work on sized types, the type system introduced in the second part of this paper is not a significant advance, but as mentioned before this is not the point of the present work. In order to justify the correctness of systems with sized types, usually strong normalization on typable terms is shown for a restriction of the reduction relation. We provide an infinitary rewriting semantics. Our approach may probably be extended to provide an infinitary rewriting semantics for at least some of the systems from the type theory literature. This semantics is interesting in its own right.

In [43] infinitary weak normalization is proven for a broad class of Pure Type Systems extended with corecursion on streams (CoPTSs), which includes Krishnaswami and Benton's typed λ -calculus of reactive programs [36]. This is related to our work in that it provides some infinitary rewriting interpretation for a class of type systems. The formalism of CoPTSs is

not based on sized types, but on a modal *next* operator, and it only supports the coinductive type of streams.

Our work is also related to the work on computability at higher types [37], but we have not yet investigated the precise relationships.

Coinduction has been studied from a more general coalgebraic perspective [26]. In this paper we use a few simple proofs by coinduction and one definition by corecursion. Formally, they could be justified as in e.g. [35, 39, 26, 11]. Our use of coinduction in this paper is not very involved, and there are no implicit corecursive function definitions like in [11].

2. INFINITARY REWRITING

In this section we define infinitary terms and reductions. We assume familiarity with the lambda calculus [5] and basic notions such as α -conversion, substitution, etc. Prior familiarity with infinitary rewriting or infinitary lambda calculus [27, 30] is not necessary but is helpful.

We assume a countable set \mathcal{V} of *variables*, and a countable set \mathcal{C} of *constructors*. The set \mathbb{T}^∞ of all finite and infinite *terms* t is given by

$$t ::= x \mid c \mid \lambda x.t \mid tt \mid \text{case}(t; \{c_k \vec{x} \Rightarrow t_k \mid k = 1, \dots, n\})$$

where $x \in \mathcal{V}$ and $c, c_k \in \mathcal{C}$. We use the notation \vec{t} (resp. \vec{x}) to denote a sequence of terms (resp. variables) of an unspecified length.

More precisely, the set \mathbb{T}^∞ is defined as an appropriate metric completion (analogously to [27]), but the above specification is clear and the details of the definition are not significant for our purposes. We consider terms modulo α -conversion. Below (Definition 2.7) we will present the terms together with the rewrite rules as an iCRS [34], which may be considered a formal definition of our rewrite system.

There are the following reductions:

$$\begin{array}{ll} (\lambda x.t)t' & \rightarrow_\beta t[t'/x] \\ \text{case}(c_k \vec{u}; \{c_l \vec{x} \Rightarrow t_l\}) & \rightarrow_\iota t_k[\vec{u}/\vec{x}] \end{array}$$

In the ι -rule we require that the appropriate sequences \vec{u} and \vec{x} have the same lengths, all variables in each \vec{x} are pairwise distinct, and the constructors c_l are all distinct. For instance, $\text{case}(ct_1t_2; \{cxy \Rightarrow x, dxy \Rightarrow y\}) \rightarrow_\iota t_1$ (assuming $c \neq d$), but $\text{case}(ct_1; \{cxy \Rightarrow x, dxy \Rightarrow y\})$, $\text{case}(c't_1t_2; \{cxy \Rightarrow x, dxy \Rightarrow y\})$ and $\text{case}(ct_1t_2; \{cxy \Rightarrow x, cxy \Rightarrow y\})$ do not have ι -reducts (assuming $c' \notin \{c, d\}$). We usually write $t \rightarrow^* t'$ to denote a finitary reduction $t \rightarrow_{\beta\iota}^* t'$.

Definition 2.1. Following [20, 17, 18], we define infinitary reduction $t \rightarrow^\infty t'$ coinductively.

$$\begin{array}{c} \frac{t \rightarrow^* x}{t \rightarrow^\infty x} \quad \frac{t \rightarrow^* c}{t \rightarrow^\infty c} \\ \\ \frac{t \rightarrow^* \lambda x.r \quad r \rightarrow^\infty r'}{t \rightarrow^\infty \lambda x.r'} \quad \frac{t \rightarrow^* r_1r_2 \quad r_k \rightarrow^\infty r'_k}{t \rightarrow^\infty r'_1r'_2} \\ \\ \frac{t \rightarrow^* \text{case}(r; \{c_k \vec{x} \Rightarrow r_k\}) \quad r \rightarrow^\infty r' \quad r_k \rightarrow^\infty r'_k}{t \rightarrow^\infty \text{case}(r'; \{c_k \vec{x} \Rightarrow r'_k\})} \end{array}$$

Intuitively, $t \rightarrow^\infty t'$ holds if it may be obtained as the conclusion of a potentially infinite derivation tree built using the above rules. The idea with the definition of the infinitary reduction \rightarrow^∞ is that the depth at which a redex is contracted should tend to infinity.

This is achieved by defining \rightarrow^∞ in such a way that always after finitely many reduction steps the subsequent contractions may be performed only at a greater depth. In other words, if $t \rightarrow^\infty t'$ then to produce any finite prefix of t' only a finitary reduction from t is necessary, i.e., any finite prefix of t' becomes fixed after finitely many reduction steps and afterwards all reductions occur only at higher depths. The idea for the definition of \rightarrow^∞ comes from [20, 17, 18].

Our coinductively defined notion of infinitary reduction corresponds to the established notion of strongly convergent reduction in infinitary rewriting [27] (see Lemma 2.8). This notion has good formal properties and an intuitive computational interpretation. Note that this is different from weak (Cauchy) convergence where one requires convergence with respect to the metric topology on terms, but the depth of the reduction activity is not required to increase. A reduction sequence may weakly converge to a limit, even though every step is performed at the root. The term can then be thought of as still changing, even though in the limit it is being reduced to itself. See [27, Section 12.3] for a more detailed discussion.

The proofs of the next three lemmas follow the pattern from [20, Lemma 4.3-4.5].

Lemma 2.2. *If $t_1 \rightarrow^\infty t'_1$ and $t_2 \rightarrow^\infty t'_2$ then $t_1[t_2/x] \rightarrow^\infty t'_1[t'_2/x]$.*

Proof. Coinduction with case analysis on $t_1 \rightarrow^\infty t'_1$, using that $t \rightarrow^* t'$ implies $t[t_2/x] \rightarrow^* t'[t_2/x]$. \square

Lemma 2.3. *If $t \rightarrow^\infty t' \rightarrow_{\beta_t} t''$ then $t \rightarrow^\infty t''$.*

Proof. Induction on $t' \rightarrow_{\beta_t} t''$, using Lemma 2.2. \square

Lemma 2.4. *If $t \rightarrow^\infty t' \rightarrow^\infty t''$ then $t \rightarrow^\infty t''$.*

Proof. By coinduction, analyzing $t' \rightarrow^\infty t''$ and using Lemma 2.3. \square

The rest of this section contains some technical definitions and results which are needed for the proof of the approximation theorem. A reader not interested in the infinitary rewriting details of this proof may skip the remainder of this section.

Definition 2.5. We define the relation $\rightarrow^{2\infty}$ analogously to \rightarrow^∞ , but replacing \rightarrow^* with \rightarrow^∞ and \rightarrow^∞ with $\rightarrow^{2\infty}$ in Definition 2.1.

We may consider \rightarrow^∞ (resp. $\rightarrow^{2\infty}$) as defining a strongly convergent ordinal-indexed reduction sequence [27] of length at most ω (resp. ω^2), obtained by concatenating the finite reductions \rightarrow^* occurring in the coinductive derivation. The next lemma may be seen as a kind of compression lemma.

Lemma 2.6. *If $t \rightarrow^{2\infty} t'$ then $t \rightarrow^\infty t'$.*

Proof. By coinduction, using Lemma 2.4. See for example [11, Lemma 6.3] for details. \square

The system of β_t -reductions on infinitary terms \mathbb{T}^∞ may be presented as a fully-extended infinitary Combinatory Reduction System (iCRS) [34]. One checks that this iCRS is orthogonal. A reader not familiar with the iCRS formalism may skip the following definition.

Definition 2.7. The signature of the iCRS contains:

- a distinct nullary symbol c for each constructor,
- a binary symbol app denoting application,
- a unary symbol lam denoting lambda abstraction, and

- for each $n \in \mathbb{N}$ and each sequence of distinct constructors c_1, \dots, c_n and each sequence of natural numbers k_1, \dots, k_n , a symbol $\text{case}_{c_1, \dots, c_n}^{k_1, \dots, k_n}$ of arity $n + 1$.

The iCRS has the following rewrite rules:

- $\text{app}(\text{lam}([x]Z(x)), X) \rightarrow Z(X)$,
- for each symbol $\text{case}_{c_1, \dots, c_n}^{k_1, \dots, k_n}$ and each $i = 1, \dots, n$:

$$\begin{aligned} & \text{case}_{c_1, \dots, c_n}^{k_1, \dots, k_n}(\text{app}(\dots(\text{app}(\text{app}(c_i, X_1), X_2) \dots), X_{k_i}), \\ & \quad [x_1, \dots, x_{k_1}]Z_1(x_1, \dots, x_{k_1}), \dots, [x_1, \dots, x_{k_n}]Z_n(x_1, \dots, x_{k_n})) \\ & \rightarrow \\ & Z_i(X_1, \dots, X_{k_i}) \end{aligned}$$

We assume x_1, \dots, x_{k_i} to be pairwise distinct, for $i = 1, \dots, n$.

One sees that this iCRS corresponds to our informal presentation of terms and reductions, and that it is fully-extended and orthogonal.

Our coinductive definition of the infinitary reduction relation \rightarrow^∞ corresponds to, in the sense of existence, to the well-established notion of strongly convergent reduction sequences [34, 27]. This is made precise in the next lemma.

Lemma 2.8. *$t \rightarrow^\infty t'$ iff there exists a strongly convergent reduction sequence from t to t' .*

Proof. This follows by a proof completely analogous to [11, Theorem 6.4], [10, Theorem 48] or [20, Theorem 3]. The technique originates from [20]. Lemma 2.6 is needed in the proof. \square

Definition 2.9. A term t is *root-active* if for every t' with $t \rightarrow^\infty t'$ there is a $\beta\mathcal{U}$ -redex t'' such that $t' \rightarrow^\infty t''$. The set of root-active, or *meaningless*, terms is denoted by \mathcal{U} . By $\sim_{\mathcal{U}}$ we denote equality of terms modulo equivalence of meaningless subterms.

Meaningless terms are a technical notion needed in the proofs, because for infinitary rewriting confluence holds only modulo $\sim_{\mathcal{U}}$. Intuitively, meaningless terms have no “meaningful” interpretation and may all be identified. An example of a meaningless term is $\Omega = (\lambda x.xx)(\lambda x.xx)$. Various other sets of meaningless terms have been considered in the infinitary lambda calculus [27, 13, 41, 42, 40, 31]. The set of root-active terms is a subset of each of them.

Because our iCRS is fully-extended and orthogonal, the following are consequences of some results in [32] and the previous lemma. Note that because all rules are collapsing, in our setting root-active terms are the same as the hypercollapsing terms from [32].

Lemma 2.10. *If $t \sim_{\mathcal{U}} t' \sim_{\mathcal{U}} t''$ then $t \sim_{\mathcal{U}} t''$.*

Proof. Follows from [32, Proposition 4.12]. \square

Lemma 2.11. *If $t \rightarrow^\infty w$ and $t \sim_{\mathcal{U}} t'$ then there is w' with $t' \rightarrow^\infty w'$ and $w \sim_{\mathcal{U}} w'$.*

Proof. Follows from [32, Lemma 4.14]. \square

Theorem 2.12. *The relation of infinitary reduction \rightarrow^∞ is confluent modulo \mathcal{U} , i.e., if $t \sim_{\mathcal{U}} t'$ and $t \rightarrow^\infty u$ and $t' \rightarrow^\infty u'$ then there exist w, w' such that $w \sim_{\mathcal{U}} w'$ and $u \rightarrow^\infty w$ and $u' \rightarrow^\infty w'$.*

Proof. Follows from [32, Theorem 4.17]. \square

3. TYPES

In this section we define the types for which we will provide an interpretation in our rewriting semantics. Some types will be decorated with sizes of (co)inductive types, indicating the type of approximations of a (co)inductive type of a given size.

Definition 3.1. *Size expressions* are given by the following grammar:

$$s ::= \infty \mid 0 \mid i \mid s + 1 \mid \min(s, s) \mid \max(s, s)$$

where i is a size variable. We denote the set of size variables by \mathcal{V}_S .

We use obvious abbreviations for size expressions, e.g., $i + 3$ for $((i + 1) + 1) + 1$, or $\min(s_1, s_2, s_3)$ for $\min(\min(s_1, s_2), s_3)$, or $\max(s)$ for s , etc. Substitution $s[s'/i]$ of s' for the size variable i in the size expression s is defined in the obvious way.

Definition 3.2. We assume an infinite set \mathcal{D} of (co)inductive definition names d, d', d_1, \dots . Types τ, α, β are defined by:

$$\tau ::= A \mid d^s(\tau_1, \dots, \tau_n) \mid \tau_1 \rightarrow \tau_2 \mid \forall i. \tau$$

where $A \in \mathcal{V}_T$ is a type variable, s is a size expression, i is a size variable, and d is a (co)inductive definition name.

A type τ is *strictly positive* if one of the following holds:

- τ is closed (i.e. it contains no type variables),
- $\tau = A$ is a type variable,
- $\tau = \tau_1 \rightarrow \tau_2$ and τ_1 is closed and τ_2 is strictly positive,
- $\tau = \forall i. \tau'$ and τ' is strictly positive,
- $\tau = d^\infty(\vec{\alpha})$ and each α_k is strictly positive.

By $\text{SV}(s)$ (resp. $\text{SV}(\tau)$) we denote the set of all size variables occurring in s (resp. τ). By $\text{TV}(\tau)$ we denote the set of all type variables occurring in τ . By $\text{FSV}(\tau)$ we denote the set of all free size variables occurring in τ (i.e. those not bound by any \forall).

Substitution $\tau[\tau'/A]$, $s[s'/i]$, $\tau[s'/i]$ is defined in the obvious way, avoiding size variable capture. We abbreviate simultaneous substitution $\tau[\alpha_1/A_1, \dots, \alpha_n/A_n]$ to $\tau[\vec{\alpha}/\vec{A}]$.

To each (co)inductive definition name $d \in \mathcal{D}$ we associate a unique (co)inductive definition. Henceforth, we will use (co)inductive definitions and their names interchangeably. Remember, however, that strictly speaking (co)inductive definitions do not occur in types, only their names do.

Definition 3.3. A *coinductive definition* for $d \in \mathcal{D}$ is specified by a defining equation of the form

$$d(B_1, \dots, B_n) = \text{CoInd}(A)\{c_k : \vec{\sigma}_k \mid k = 1, \dots, m\}$$

where A is the *recursive type variable*, and B_1, \dots, B_n are the *parameter type variables*, and $m > 0$, and c_k is the k th *constructor*, and σ_k^l is the k th constructor's l th *argument type*, and the following is satisfied:

- σ_k^l are all strictly positive,
- $\text{TV}(\sigma_k^l) \subseteq \{A, B_1, \dots, B_n\}$,
- $\text{FSV}(\sigma_k^l) = \emptyset$.

An *inductive definition* is specified analogously, but using Ind instead of CoInd .

We assume that each constructor c is associated with a unique (co)inductive definition $\text{Def}(c)$.

We assume there is a well-founded order \prec on (co)inductive definitions such that for every (co)inductive definition d , each (co)inductive definition d' occurring in a constructor argument type of d satisfies $d' \prec d$.

The type variable A is used as a placeholder for recursive occurrences of $d(\vec{B})$. We often write $\text{ArgTypes}(c_k)$ to denote $(\sigma_k^1, \dots, \sigma_k^{n_k})$: the argument types of the k -th constructor. We usually present (co)inductive definitions in a bit more readable format by replacing the recursive type variable A with the type being defined, presenting the constructor argument types in a chain of arrow types, and adding the type being defined as the target type of constructors. For instance, the inductive definition of lists is specified by

$$\text{List}(B) = \text{Ind}\{\text{nil} : \text{List}(B), \text{cons} : B \rightarrow \text{List}(B) \rightarrow \text{List}(B)\}.$$

Formally, here $\sigma_1^1 = A$, $\sigma_2^1 = B$, and $\sigma_2^2 = A$.

Example 3.4. The inductive definition of natural numbers is specified by:

$$\text{Nat} = \text{Ind}\{0 : \text{Nat}, S : \text{Nat} \rightarrow \text{Nat}\}.$$

The coinductive definition of streams of natural numbers is specified by:

$$\text{Strm} = \text{CoInd}\{\text{cons} : \text{Nat} \rightarrow \text{Strm} \rightarrow \text{Strm}\}.$$

Definition 3.5. An expression of the form $d(\tau_1, \dots, \tau_n)$ is a *(co)inductive type*, depending on whether d is an inductive or coinductive definition. A type of the form $d^s(\tau_1, \dots, \tau_n)$ is a *decorated (co)inductive type*. We drop the designator “decorated” when clear from the context. We write $c \in \text{Constr}(\rho)$ to denote that c is a constructor for a (decorated) (co)inductive type or definition ρ .

In a (co)inductive type $d^s(\tau_1, \dots, \tau_n)$, the types τ_1, \dots, τ_n denote the *parameters*. Intuitively, we substitute τ_1, \dots, τ_n for the parameter type variables B_1, \dots, B_n of the (co)inductive definition d .

By default, d_ν denotes a coinductive and d_μ an inductive definition. We use μ for inductive and ν for coinductive types, and ρ for (co)inductive types when it is not important if it is inductive or coinductive. Analogously, we use μ^s, ν^s, ρ^s for decorated (co)inductive types (with size s). We often omit the superscript ∞ in ρ^∞ , overloading the notation.

Intuitively, μ^s denotes the type of objects of an inductive type μ which have size at most s , and ν^s denotes the type of objects of a coinductive type ν which have size at least s , i.e., considered up to depth s they represent a valid object of type ν . For a stream $\nu = \text{Strm}$, the type Strm^s is the type of terms t which produce (under a sufficiently long reduction sequence) at least s initial elements of a stream. The type e.g. $\forall i. \text{Strm}^i \rightarrow \text{Strm}^s$ is the type of functions which when given as argument a stream of size i (i.e. with at least i initial elements well-defined) produce at least s initial elements of a stream, where i may occur in s .

Note that the parameters to (co)inductive definitions may be other (co)inductive types with size constraints. For instance $\text{List}(\text{List}^i(\tau))$ denotes the type of lists (of any length) whose elements are lists of length at most i with elements of type τ . Note also that the recursive type variable A may occur as a parameter of a (co)inductive type in the type of one of the constructors. For these two reasons we need to require that the parameter type variables occur only strictly positively in the types of the arguments of constructors. One could allow non-positive occurrences of parameter type variables in general and restrict the occurrences to strictly positive only for instantiations with types containing free size

variables or recursive type variables. This would, however, introduce some tedious but straightforward technicalities in the proofs.

Example 3.6. Infinite binary trees storing natural numbers in nodes may be specified by:

$$\text{BTree} = \text{CoInd}\{\text{bnode} : \text{Nat} \rightarrow \text{BTree} \rightarrow \text{BTree} \rightarrow \text{BTree}\}.$$

Trees with potentially infinite branches but finite branching at each node are specified by:

$$\text{FTree} = \text{CoInd}\{\text{fnode} : \text{Nat} \rightarrow \text{List}(\text{FTree}) \rightarrow \text{FTree}\}.$$

Here the type FTree itself (formally, the recursive type variable A) occurs as a parameter of List in the type of the constructor `fnode`.

Infinite trees with infinite branching are specified by:

$$\text{Tree} = \text{CoInd}\{\text{node} : \text{Nat} \rightarrow (\text{Nat} \rightarrow \text{Tree}) \rightarrow \text{Tree}\}.$$

Here infinite branching is specified by a function from Nat to Tree.

Recall the coinductive definition of the type Tree_2 from the introduction:

$$\text{Tree}_2 = \text{CoInd}\{c_1 : \text{List}(\text{Tree}_2) \rightarrow \text{Tree}_2, c_2 : (\text{Nat} \rightarrow \text{Tree}_2) \rightarrow \text{Tree}_2\}.$$

In this definition both finite branching via the c_1 constructor and infinite branching via c_2 are possible. In contrast to BTree, FTree and Tree, the nodes of Tree_2 do not store any natural number values.

Example 3.7. As an example of a nested higher-order (co)inductive type we consider stream processors from [23]. See also [3, Section 2.3]. We define two types:

$$\begin{aligned} \text{SPi}(B) &= \text{Ind}\{\text{get} : (\text{Nat} \rightarrow \text{SPi}(B)) \rightarrow \text{SPi}(B), \\ &\quad \text{put} : \text{Nat} \rightarrow B \rightarrow \text{SPi}(B)\} \\ \text{SP} &= \text{CoInd}\{\text{out} : \text{SPi}(\text{SP}) \rightarrow \text{SP}\} \end{aligned}$$

The type SP is a type of stream processors. A stream processor can either read the first element from the input stream and enter a new state depending on the read value (the `get` constructor), or it can write an element to the output stream and enter a new state (the `put` constructor). To ensure productivity, a stream processor may read only finitely many elements from the input stream before writing a value to the output stream. This is achieved by nesting the inductive type SPi inside the coinductive type SP of stream processors.

The well-founded order \prec on (co)inductive definitions essentially disallows mutual (co)inductive types. They may still be represented indirectly thanks to type parameters.

Example 3.8. The types Odd and Even of odd and even natural numbers may be defined as mutual inductive types:

$$\begin{aligned} \text{Odd} &= \text{Ind}\{S_o : \text{Even} \rightarrow \text{Odd}\} \\ \text{Even} &= \text{Ind}\{0 : \text{Even}, S_e : \text{Odd} \rightarrow \text{Even}\} \end{aligned}$$

These are not valid inductive definitions in our formalism, but they may be reformulated as follows:

$$\begin{aligned} \text{Odd}_0(B) &= \text{Ind}\{S_o : B \rightarrow \text{Odd}_0(B)\} \\ \text{Even} &= \text{Ind}\{0 : \text{Even}, S_e : \text{Odd}_0(\text{Even}) \rightarrow \text{Even}\} \end{aligned}$$

Now the type Odd is represented by $\text{Odd}_0(\text{Even})$.

In the rest of this paper by “induction on a type τ ” we mean induction on the lexicographic product of:

- the multiset extension of the well-founded order \prec on (co)inductive definitions occurring in the type, and
- the size of the type.

In this order, if $c \in \text{Constr}(\rho)$ with $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_m)$ then each σ_k is smaller than ρ .

4. REWRITING SEMANTICS

In this section we define our rewriting semantics. More precisely, we define an interpretation $\llbracket \tau \rrbracket \subseteq \mathbb{T}^\infty$ for each type τ .

By ∞ we denote a sufficiently large ordinal (see Definition 4.1), and by Ω we denote the set of all ordinals not greater than ∞ . A *size variable valuation* is a function $v : \mathcal{V}_S \rightarrow \Omega$. Any size variable valuation v extends in a natural way to a function from size expressions to Ω . More precisely, we define: $v(\infty) = \infty$, $v(0) = 0$, $v(s+1) = \min(v(s) + 1, \infty)$, $v(\min(s_1, s_2)) = \min(v(s_1), v(s_2))$, $v(\max(s_1, s_2)) = \max(v(s_1), v(s_2))$. To save on notation we identify ordinals larger than ∞ with ∞ , e.g., $\infty + 1$ denotes the ordinal ∞ .

Definition 4.1. We interpret types as subsets of \mathbb{T}^∞ . By ∞ we denote an ordinal large enough so that any monotone function on $\mathcal{P}(\mathbb{T}^\infty)$ (the powerset of \mathbb{T}^∞) reaches its least and greatest fixpoint in ∞ iterations. This ordinal exists, as we may take any ordinal larger than the cardinality of $\mathcal{P}(\mathbb{T}^\infty)$.

Given a *type variable valuation* $\xi : \mathcal{V}_T \rightarrow \mathcal{P}(\mathbb{T}^\infty)$, a size variable valuation $v : \mathcal{V}_S \rightarrow \Omega$, and a strictly positive type τ , we define a *type valuation* $\llbracket \tau \rrbracket_{\xi, v} \subseteq \mathbb{T}^\infty$. This is done by induction on τ . We simultaneously also define valuation approximations $\llbracket \rho \rrbracket_{\xi, v}^\varkappa$ and $\llbracket d \rrbracket_{\xi, v}^\varkappa$.

- Let $d(B_1, \dots, B_n) = (\text{Co})\text{Ind}(A)\{c_k : \vec{\sigma}_k \rightarrow d(\vec{B})\}$ be a (co)inductive definition. We define a function $\Phi_{d, \xi, v} : \mathcal{P}(\mathbb{T}^\infty) \rightarrow \mathcal{P}(\mathbb{T}^\infty)$ so that $\Phi_{d, \xi, v}(X)$ for $X \subseteq \mathbb{T}^\infty$ contains all terms of the form $c_k t_k^1 \dots t_k^{n_k}$ such that $t_k^l \in \llbracket \sigma_k^l \rrbracket_{\xi[X/A], v}$ for $l = 1, \dots, n_k$.

For a coinductive definition d_ν and an ordinal $\varkappa \in \Omega$ we define the *valuation approximation* $\llbracket d_\nu \rrbracket_{\xi, v}^\varkappa \subseteq \mathbb{T}^\infty$ as follows:

- $\llbracket d_\nu \rrbracket_{\xi, v}^0 = \mathbb{T}^\infty$,
- $\llbracket d_\nu \rrbracket_{\xi, v}^{\varkappa+1} = \Phi_{d_\nu, \xi, v}(\llbracket d_\nu \rrbracket_{\xi, v}^\varkappa)$,
- $\llbracket d_\nu \rrbracket_{\xi, v}^\varkappa = \bigcap_{\varkappa' < \varkappa} \llbracket d_\nu \rrbracket_{\xi, v}^{\varkappa'}$ if \varkappa is a limit ordinal.

For an inductive definition d_μ and an ordinal $\varkappa \in \Omega$ we define the *valuation approximation* $\llbracket d_\mu \rrbracket_{\xi, v}^\varkappa \subseteq \mathbb{T}^\infty$ by:

- $\llbracket d_\mu \rrbracket_{\xi, v}^0 = \emptyset$,
- $\llbracket d_\mu \rrbracket_{\xi, v}^{\varkappa+1} = \Phi_{d_\mu, \xi, v}(\llbracket d_\mu \rrbracket_{\xi, v}^\varkappa)$,
- $\llbracket d_\mu \rrbracket_{\xi, v}^\varkappa = \bigcup_{\varkappa' < \varkappa} \llbracket d_\mu \rrbracket_{\xi, v}^{\varkappa'}$ if \varkappa is a limit ordinal.

- $\llbracket \rho \rrbracket_{\xi, v}^\varkappa = \llbracket d \rrbracket_{\xi[\vec{Y}/\vec{B}], v}^\varkappa$ where $\rho = d(\vec{\alpha})$ is a (co)inductive type, $Y_j = \llbracket \alpha_j \rrbracket_{\xi, v}$, and \vec{B} are the parameter type variables of d .
- $\llbracket \rho^s \rrbracket_{\xi, v} = \llbracket \rho \rrbracket_{\xi, v}^{v(s)}$.
- $\llbracket A \rrbracket_{\xi, v} = \xi(A)$.
- $t \in \llbracket \forall i. \tau \rrbracket_{\xi, v}$ if $i \notin \text{FSV}(t)$ and for every $\varkappa \in \Omega$ there is t' with $t \rightarrow^\infty t' \in \llbracket \tau \rrbracket_{\xi, v[\varkappa/i]}$.
- $t \in \llbracket \alpha \rightarrow \beta \rrbracket_{\xi, v}$ if for every $r \in \llbracket \alpha \rrbracket_{\xi, v}$ there is t' with $tr \rightarrow^\infty t' \in \llbracket \beta \rrbracket_{\xi, v}$.

For a closed type τ the valuation $\llbracket \tau \rrbracket_{\xi, v}$ does not depend on ξ , so we simply write $\llbracket \tau \rrbracket_v$ instead. Whenever we omit the type variable valuation we implicitly assume the type to be closed.

In general, the interpretation $\llbracket \tau \rrbracket$ of a type τ may contain terms which are not in normal form. This is because of the interpretation of function types and quantification over size variables ($\forall i$). If τ is a simple first-order (co)inductive type whose constructor argument types contain neither function types ($\tau_1 \rightarrow \tau_2$) nor quantification over size variables ($\forall i. \tau'$), then $\llbracket \tau \rrbracket$ contains only normal forms.

Thus, we do not show infinitary weak normalization for terms having function types. Nonetheless, our interpretation of $t \in \llbracket \tau_1 \rightarrow \tau_2 \rrbracket$ is very natural and ensures productivity of t regarded as a function: we require that for $u \in \llbracket \tau_1 \rrbracket$ there is $u' \in \llbracket \tau_2 \rrbracket$ with $tu \rightarrow^\infty u'$. Intuitively, this means that for any $u \in \llbracket \tau_1 \rrbracket$ the application tu reduces “in the limit” to a term $u' \in \llbracket \tau_2 \rrbracket$, using only a finite initial part of u to produce a finite initial part of u' . Moreover, it is questionable in the first place how sensible infinitary normalization is as a “correctness” criterion for terms of function types.

Example 4.2. Recall the definitions of the types Nat and Strm from Example 3.4:

$$\begin{aligned} \text{Nat} &= \text{Ind}\{0 : \text{Nat}, S : \text{Nat} \rightarrow \text{Nat}\} \\ \text{Strm} &= \text{CoInd}\{\text{cons} : \text{Nat} \rightarrow \text{Strm} \rightarrow \text{Strm}\} \end{aligned}$$

The elements of $\llbracket \text{Nat} \rrbracket$ are the terms: $0, S(0), S(S(0)), \dots$. We use common number notation, e.g. 1 for $S(0)$, etc. We usually write e.g. $1 :: 2 :: t$ instead of $\text{cons } 1 (\text{cons } 2 t)$. The elements of $\llbracket \text{Strm} \rrbracket$ are all infinite terms of the form $n_1 :: n_2 :: n_3 :: \dots$ where $n_i \in \llbracket \text{Nat} \rrbracket$.

Consider the term

$$\mathbf{t1} = \lambda t. \text{case}(t; \{\text{cons } x y \Rightarrow y\})$$

We have $\mathbf{t1} \in \llbracket \text{Strm} \rightarrow \text{Strm} \rrbracket$. Indeed, let $t \in \llbracket \text{Strm} \rrbracket$. Then $t = n :: t'$ with $n \in \llbracket \text{Nat} \rrbracket$ and $t' \in \llbracket \text{Strm} \rrbracket$. Thus $\mathbf{t1}(t) \rightarrow \text{case}(n :: t'; \{\text{cons } x y \Rightarrow y\}) \rightarrow t' \in \llbracket \text{Strm} \rrbracket$.

Example 4.3. Recall the definitions of BTree , FTree and Tree from Example 3.6:

$$\begin{aligned} \text{BTree} &= \text{CoInd}\{\mathbf{bnode} : \text{Nat} \rightarrow \text{BTree} \rightarrow \text{BTree} \rightarrow \text{BTree}\} \\ \text{FTree} &= \text{CoInd}\{\mathbf{fnode} : \text{Nat} \rightarrow \text{List}(\text{FTree}) \rightarrow \text{FTree}\} \\ \text{Tree} &= \text{CoInd}\{\mathbf{node} : \text{Nat} \rightarrow (\text{Nat} \rightarrow \text{Tree}) \rightarrow \text{Tree}\} \end{aligned}$$

The interpretation $\llbracket \text{BTree} \rrbracket$ consists of all infinite terms of the form

$$\mathbf{bnode } n_{1,1} (\mathbf{bnode } n_{2,1} (\dots) (\dots)) (\mathbf{bnode } n_{2,2} (\dots) (\dots))$$

where $n_{1,1}, n_{2,1}, n_{2,2}, \dots \in \llbracket \text{Nat} \rrbracket$. The interpretation $\llbracket \text{FTree} \rrbracket$ consists of all potentially infinite terms of the form $\mathbf{fnode } n_{1,1} ((\mathbf{fnode } n_{2,1} (\dots)) :: (\mathbf{fnode } n_{2,2} (\dots)) :: \dots :: \text{nil})$ where $n_{1,1}, n_{2,1}, n_{2,2}, \dots \in \llbracket \text{Nat} \rrbracket$. Finally, $\llbracket \text{Tree} \rrbracket$ consists of all terms of the form $\mathbf{node } n f$ where $n \in \llbracket \text{Nat} \rrbracket$ for every $m \in \llbracket \text{Nat} \rrbracket$ there is $t \in \llbracket \text{Tree} \rrbracket$ such that $fm \rightarrow^\infty t$.

Example 4.4. Recall the definition of stream processors from Example 3.7:

$$\begin{aligned} \text{SPi}(B) &= \text{Ind}\{\mathbf{get} : (\text{Nat} \rightarrow \text{SPi}(B)) \rightarrow \text{SPi}(B), \\ &\quad \mathbf{put} : \text{Nat} \rightarrow B \rightarrow \text{SPi}(B)\} \\ \text{SP} &= \text{CoInd}\{\mathbf{out} : \text{SPi}(\text{SP}) \rightarrow \text{SP}\} \end{aligned}$$

An example stream processor, i.e., an example element of $\llbracket \text{SP} \rrbracket$ is an infinite term odd satisfying the identity:

$$\text{odd} = \text{out}(\mathbf{get}(\lambda x. \mathbf{get}(\lambda y. \mathbf{put } x \text{ odd})))$$

The stream processor `odd` drops every second element of a stream, e.g., it transforms the stream $1 :: 2 :: 3 :: 4 :: \dots$ into $1 :: 3 :: 5 :: \dots$. But e.g. the infinite term

$$\text{out}(\text{get}(\lambda x_1.\text{get}(\lambda x_2.\text{get}(\lambda x_3.\text{get}(\dots))))))$$

is *not* in $\llbracket \text{SP} \rrbracket$, because it nests infinitely many `gets`.

Lemma 4.5. *If $v(i) = v'(i)$ for every $i \in \text{FSV}(\tau)$ then $\llbracket \tau \rrbracket_{\xi,v} = \llbracket \tau \rrbracket_{\xi,v'}$. Moreover, $\llbracket d \rrbracket_{\xi,v}^{\varkappa} = \llbracket d \rrbracket_{\xi,v'}^{\varkappa}$ for any v, v' .*

Proof. Follows by induction on τ , using the fact $\text{FSV}(\sigma_k^l) = \emptyset$ for σ_k^l a constructor argument type as in Definition 3.3. \square

Lemma 4.6.

- (1) *If $\xi(A) = \xi'(A)$ for $A \in \text{TV}(\tau)$ then $\llbracket \tau \rrbracket_{\xi,v} = \llbracket \tau \rrbracket_{\xi',v}$.*
- (2) *If $\xi(B_i) = \xi'(B_i)$ for each parameter type variable B_i of d , then $\llbracket d \rrbracket_{\xi,v}^{\varkappa} = \llbracket d \rrbracket_{\xi',v}^{\varkappa}$.*

Proof. Induction on τ , generalizing over ξ, ξ' and v . \square

Corollary 4.7. *If $\xi(B_i) = \xi'(B_i)$ for each parameter type variable B_i of d , then $\Phi_{d,\xi,v} = \Phi_{d,\xi',v}$.*

Lemma 4.8. *Assume $\xi \subseteq \xi'$, i.e., $\xi(A) \subseteq \xi'(A)$ for all type variables A .*

- (1) *If τ is strictly positive then $\llbracket \tau \rrbracket_{\xi,v} \subseteq \llbracket \tau \rrbracket_{\xi',v}$.*
- (2) *If d is a (co)inductive definition then $\llbracket d \rrbracket_{\xi,v}^{\varkappa} \subseteq \llbracket d \rrbracket_{\xi',v}^{\varkappa}$.*
- (3) *If $X \subseteq X'$ then $\Phi_{d,\xi,v}(X) \subseteq \Phi_{d,\xi',v}(X')$. In particular, the function $\Phi_{d,\xi,v}$ is monotone.*

Proof. Induction on τ , generalizing over ξ, ξ', v . \square

From the third point in the above lemma it follows that $\llbracket d_\nu \rrbracket_{\xi,v}^{\varkappa_1} \subseteq \llbracket d_\nu \rrbracket_{\xi,v}^{\varkappa_2}$ for $\varkappa_2 \leq \varkappa_1$, and $\llbracket d_\mu \rrbracket_{\xi,v}^{\varkappa_1} \subseteq \llbracket d_\mu \rrbracket_{\xi,v}^{\varkappa_2}$ for $\varkappa_1 \leq \varkappa_2$. Also, for a (co)inductive definition d , by the Knaster-Tarski fixpoint theorem [45], the function $\Phi_{d,\xi,v}$ has the least and greatest fixpoints, which may be obtained by “iterating” $\Phi_{d,\xi,v}$ starting with the empty or the full set, respectively, as in the definition of valuation approximations. For an inductive definition d_μ , the least fixpoint of $\Phi_{d_\mu,\xi,v}$ is then $\llbracket d_\mu \rrbracket_{\xi,v}^\infty$, by how we defined ∞ . Analogously, for a coinductive definition d_ν the greatest fixpoint of $\Phi_{d_\nu,\xi,v}$ is $\llbracket d_\nu \rrbracket_{\xi,v}^\infty$. Note that for $\varkappa \geq \infty$ we have $\llbracket d \rrbracket_{\xi,v}^{\varkappa} = \llbracket d \rrbracket_{\xi,v}^\infty$.

The next definition and the ensuing lemma are needed in the proof of the approximation theorem. A reader not interested in the details of this proof may skip the rest of this section.

Definition 4.9. A set $X \subseteq \mathbb{T}^\infty$ is *stable* when:

- (1) if $t \in X$ and $t \sim_{\mathcal{U}} t'$ then $t' \in X$,
- (2) if $t \in X$ and $t \rightarrow^\infty t'$ then $t' \in X$.

A type variable valuation ξ is *stable* if $\xi(A)$ is stable for each type variable A . The following lemma implies that the interpretations of closed types are in fact stable.

Lemma 4.10. *Assume τ, ρ are strictly positive.*

- (1) *If ξ is stable then so is $\llbracket \tau \rrbracket_{\xi,v}$.*
- (2) *If ξ is stable then so is $\llbracket \rho \rrbracket_{\xi,v}^{\varkappa}$.*
- (3) *If ξ and $X \subseteq \mathbb{T}^\infty$ are stable then so is $\Phi_{d_\rho,\xi,v}(X)$.*

Proof. We show the first point by induction on τ , generalizing over ξ, v . The remaining two points will follow directly from this proof.

First assume $\tau = \rho^s$ with $\rho = d(\vec{\alpha})$. Then $\llbracket \tau \rrbracket_{\xi, v} = \llbracket \rho \rrbracket_{\xi, v}^{v(s)} = \llbracket d \rrbracket_{\xi[\vec{Y}/\vec{B}], v}^{v(s)}$ where $Y_j = \llbracket \alpha_j \rrbracket_{\xi, v}$ and each α_j is strictly positive. By the inductive hypothesis each Y_j is stable. Hence $\xi_1 = \xi[\vec{Y}/\vec{B}]$ is also stable. We show that if $X \subseteq \mathbb{T}^\infty$ is stable then so is $\Phi_{d, \xi_1, v}(X)$. From this it follows by induction that $\llbracket \rho \rrbracket_{\xi, v}^\varkappa$ is stable for any $\varkappa \in \Omega$, and thus $\llbracket \tau \rrbracket_{\xi, v}$ is stable. Let $t \in \Phi_{d, \xi_1, v}(X)$. Then $t = ct_1 \dots t_n$ where $t_k \in \llbracket \sigma_k \rrbracket_{\xi_1[X/A], v}$ and $c \in \text{Constr}(\rho)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$. Note that $\xi_1[X/A]$ is stable, because X is. Hence $\llbracket \sigma_k \rrbracket_{\xi_1[X/A], v}$ is stable by the inductive hypothesis.

(1) Assume $t \sim_{\mathcal{U}} t'$. Then $t' = ct'_1 \dots t'_n$ with $t_k \sim_{\mathcal{U}} t'_k$. We have $t'_k \in \llbracket \sigma_k \rrbracket_{\xi_1[X/A], v}$ because $\llbracket \sigma_k \rrbracket_{\xi_1[X/A], v}$ is stable. Thus $t' \in \Phi_{d, \xi_1, v}(X)$.

(2) Assume $t \rightarrow^\infty t'$. Then $t' = ct'_1 \dots t'_n$ with $t_k \rightarrow^\infty t'_k$. We have $t'_k \in \llbracket \sigma_k \rrbracket_{\xi_1[X/A], v}$ because $\llbracket \sigma_k \rrbracket_{\xi_1[X/A], v}$ is stable. Thus $t' \in \Phi_{d, \xi_1, v}(X)$.

If $\tau = A$ is a type variable then $\llbracket \tau \rrbracket_{\xi, v} = \xi(A)$ is stable because ξ is.

Assume $\tau = \forall i. \tau'$. Let $t \in \llbracket \tau \rrbracket_{\xi, v}$.

(1) Assume $t \sim_{\mathcal{U}} t'$. Let $\varkappa \in \Omega$. There is t_0 with $t \rightarrow^\infty t_0 \in \llbracket \tau' \rrbracket_{\xi, v[\varkappa/i]}$. By Lemma 2.11 there is t'_0 with $t_0 \sim_{\mathcal{U}} t'_0$ and $t' \rightarrow^\infty t'_0$. By the inductive hypothesis $\llbracket \tau' \rrbracket_{\xi, v[\varkappa/i]}$ is stable, so $t'_0 \in \llbracket \tau' \rrbracket_{\xi, v[\varkappa/i]}$. Thus $t' \in \llbracket \tau \rrbracket_{\xi, v}$ (without loss of generality $i \notin \text{FSV}(t')$).

(2) Assume $t \rightarrow^\infty t'$. There is t_0 with $t \rightarrow^\infty t_0 \in \llbracket \tau' \rrbracket_{\xi, v[\varkappa/i]}$. By confluence modulo \mathcal{U} there are t_1, t_2 with $t_0 \rightarrow^\infty t_1 \sim_{\mathcal{U}} t_2$ and $t' \rightarrow^\infty t_2$. By the inductive hypothesis $\llbracket \tau' \rrbracket_{\xi, v[\varkappa/i]}$ is stable, so $t_2 \in \llbracket \tau' \rrbracket_{\xi, v[\varkappa/i]}$. Thus $t' \in \llbracket \tau \rrbracket_{\xi, v}$.

Finally, assume $\tau = \tau_1 \rightarrow \tau_2$ with τ_1 closed and τ_2 strictly positive. Let $t \in \llbracket \tau \rrbracket_{\xi, v}$. By the inductive hypothesis $\llbracket \tau_2 \rrbracket_{\xi, v}$ is stable.

(1) Assume $t \sim_{\mathcal{U}} t'$. We need to show $t' \in \llbracket \tau \rrbracket_{\xi, v}$. Let $r \in \llbracket \tau_1 \rrbracket_{\xi, v}$. Then $tr \rightarrow^\infty t_0 \in \llbracket \tau_2 \rrbracket_{\xi, v}$. We have $tr \sim_{\mathcal{U}} t'r$, so by Lemma 2.11 there is t'_0 with $t_0 \sim_{\mathcal{U}} t'_0$ and $t'r \rightarrow^\infty t'_0$. Because $\llbracket \tau_2 \rrbracket_{\xi, v}$ is stable, $t'_0 \in \llbracket \tau_2 \rrbracket_{\xi, v}$.

(2) Assume $t \rightarrow^\infty t'$. We need to show $t' \in \llbracket \tau \rrbracket_{\xi, v}$. Let $r \in \llbracket \tau_1 \rrbracket_{\xi, v}$. Then $tr \rightarrow^\infty t_0 \in \llbracket \tau_2 \rrbracket_{\xi, v}$. We have $tr \rightarrow^\infty t'r$, so by confluence there are t_1, t_2 with $t_0 \rightarrow^\infty t_1 \sim_{\mathcal{U}} t_2$ and $t'r \rightarrow^\infty t_2$. Because $\llbracket \tau_2 \rrbracket_{\xi, v}$ is stable, $t_2 \in \llbracket \tau_2 \rrbracket_{\xi, v}$. \square

5. APPROXIMATION THEOREM

In this section we prove the approximation theorem: if $t \rightarrow^\infty t_n \in \llbracket \nu \rrbracket_v^n$ for $n \in \mathbb{N}$ then there exists $t_\infty \in \llbracket \nu \rrbracket_v^\infty$ such that $t \rightarrow^\infty t_\infty$.

The approximation theorem is an easy consequence of the following result: if $t_n \rightarrow^\infty t_{n+1}$ and $t_n \in \llbracket \nu \rrbracket_v^n$ for $n \in \mathbb{N}$, then there exists t_∞ such that $t_0 \rightarrow^\infty t_\infty \in \llbracket \nu \rrbracket_v^\infty$. If ν is a simple coinductive type, e.g., it is a stream with a single constructor c where $\text{ArgTypes}(c) = (\sigma, A)$, the type σ is closed, and A is the recursive type variable of ν , then the argument is not complicated. It follows from the assumption that $t_{n+1} = cu_{n+1}w_{n+1}$ with $u_{n+1} \in \llbracket \sigma \rrbracket_v$, $w_{n+1} \in \llbracket \nu \rrbracket_v^n$ and $w_{n+1} \rightarrow^\infty w_{n+2}$. We coinductively construct w_∞ with $w_1 \rightarrow^{2^\infty} w_\infty \in \llbracket \nu \rrbracket_v^\infty$ (note that $\llbracket \nu \rrbracket_v^\infty$ treated as a unary relation may be defined coinductively). Take $t_\infty = cu_1w_\infty$. We have $t_0 \rightarrow^{2^\infty} t_\infty \in \llbracket \nu \rrbracket_v^\infty$, which suffices by Lemma 2.6. This reasoning captures the gist

of the argument. With higher-order (co)inductive types the core idea remains the same but significant technical complications occur because of the alternation of least and greatest fixpoints in the definition of $\llbracket - \rrbracket_{\xi, v}$. We construct the term t_∞ by coinduction, and show $t_0 \rightarrow^\infty t_\infty$ by coinduction, and then show $t_\infty \in \llbracket \nu \rrbracket_v^\infty$ by an inductive argument. To be able to even state an appropriately generalized inductive hypothesis, we first need some definitions.

A reader not interested in the infinitary rewriting details of the proof of the approximation theorem may skip directly to Theorem 5.22.

Definition 5.1. Let τ be a strictly positive type and $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ a family of type variable valuations. A τ, Ξ -sequence (with v) is a sequence of terms $\{t_n\}_{n \in \mathbb{N}}$ satisfying $t_n \in \llbracket \tau \rrbracket_{\xi_n, v}$ and $t_n \rightarrow^\infty t_{n+1}$ for $n \in \mathbb{N}$.

By $\Xi'_v = \{\xi'_n\}_{n \in \mathbb{N}}$ we denote the family of type variable valuations such that $\xi'_n(A) = \llbracket \nu \rrbracket_v^n$ for all A and $n \in \mathbb{N}$. We usually write Ξ' instead of Ξ'_v when v is irrelevant or clear from the context. If $\mathcal{T} = \{\tau_A\}_{A \in V_T}$ is a family of strictly positive types and $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ a family of type variable valuations, then $\Xi \llbracket \mathcal{T} \rrbracket_v$ denotes the family $\{\xi'_n\}_{n \in \mathbb{N}}$ where $\xi'_n(A) = \llbracket \tau_A \rrbracket_{\xi_n, v}$. Again, the subscript v is usually omitted.

A family Ξ of type variable valuations is ν -hereditary (with v) if $\Xi = \Xi'_v$ or, inductively, $\Xi = \Xi' \llbracket \mathcal{T} \rrbracket_v$ for some ν -hereditary Ξ' and a family \mathcal{T} of strictly positive types.

A hereditiy derivation D is either \emptyset , or, inductively, a pair (D', \mathcal{T}) where D' is a hereditiy derivation and \mathcal{T} a family of strictly positive types. The ν -hereditary family Ξ^D determined by a hereditiy derivation D is defined inductively: $\Xi^\emptyset = \Xi^\nu$ and $\Xi^{(D', \mathcal{T})} = \Xi^D \llbracket \mathcal{T} \rrbracket$.

A family $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is *stable* if each ξ_n is stable.

For the sake of readability we usually talk about ν -hereditary families, but we always implicitly assume that for any given ν -hereditary family Ξ we are given a fixed hereditiy derivation D such that $\Xi = \Xi^D$.

Lemma 5.2. *Any ν -hereditary family Ξ is stable.*

Proof. By induction on the definition of a ν -hereditary family, using Lemma 4.10. \square

Lemma 5.3. *If a family Ξ determined by a hereditiy derivation D is ν -hereditary with v and the size variable i is fresh, i.e., it does not occur in ν or any of the types in the type families in D , then Ξ is ν -hereditary with $v[\varkappa/i]$ and determined by the same hereditiy derivation D .*

Proof. Induction on D . If $D = \emptyset$ then $\Xi = \Xi'_v = \Xi'_{v[\varkappa/i]}$ by Lemma 4.5, because i does not occur in ν . If $D = (D', \mathcal{T})$ and $\Xi = \Xi^{D'} \llbracket \mathcal{T} \rrbracket$, then by the inductive hypothesis $\Xi^{D'}$ is ν -hereditary with $v[\varkappa/i]$ and determined by the hereditiy derivation D' . Assuming $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ and $\Xi^{D'} = \{\xi'_n\}_{n \in \mathbb{N}}$, we have $\xi_n(A) = \llbracket \tau_A \rrbracket_{\xi'_n, v} = \llbracket \tau_A \rrbracket_{\xi'_n, v[\varkappa/i]}$ by Lemma 4.5 because $i \notin \text{FSV}(\tau_A)$. So Ξ is ν -hereditary with $v[\varkappa/i]$ and determined by D . \square

Lemma 5.4. *If $\{t_n\}_{n \in \mathbb{N}}$ is a A, Ξ^ν -sequence, then $t_{n+1} = ct_{n+1}^1 \dots t_{n+1}^m$ for $n \in \mathbb{N}$, and $\{t_{n+1}^k\}_{n \in \mathbb{N}}$ is a σ_k, Ξ^k -sequence for each $k = 1, \dots, m$ where $c \in \text{Constr}(\nu)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_m)$ and $\nu = d_\nu(\vec{\alpha})$ and $\Xi^k = \Xi^\nu \llbracket \mathcal{T} \rrbracket$ where $\mathcal{T} = \{\tau_{A'}\}_{A' \in V_T}$ and $\tau_{B_j} = \alpha_j$ and $\tau_{A'} = A'$ for $A' \notin \{B_1, \dots, B_l\}$ and B_1, \dots, B_l are the parameter type variables of d_ν .*

Proof. Let $\Xi' = \{\xi'_n\}_{n \in \mathbb{N}}$. We have

$$\begin{aligned} t_{n+1} &\in \llbracket A \rrbracket_{\xi_{n+1}, v}^\nu \\ &= \llbracket \nu \rrbracket_v^{n+1} \\ &= \llbracket d_\nu \rrbracket_{\xi, v}^{n+1} \\ &= \Phi_{d_\nu, \xi, v}(\llbracket d_\nu \rrbracket_{\xi, v}^n) \\ &= \Phi_{d_\nu, \xi, v}(\llbracket \nu \rrbracket_v^n) \end{aligned}$$

where $\xi(B_j) = \llbracket \alpha_j \rrbracket_v$ and B_1, \dots, B_l are the parameter type variables of d_ν . Then $t_{n+1} = c_{n+1} t_{n+1}^1 \dots t_{n+1}^{m_{n+1}}$ with $t_{n+1}^k \in \llbracket \sigma_k^{n+1} \rrbracket_{\xi[\llbracket \nu \rrbracket_v^n / A], v}$ where $\text{ArgTypes}(c_{n+1}) = (\sigma_1^{n+1}, \dots, \sigma_{m_{n+1}}^{n+1})$ and A is the recursive type variable of d_ν . Since $t_n \rightarrow^\infty t_{n+1}$ for $n \in \mathbb{N}$ we must have $c_{n+1} = c$ and $m_{n+1} = m$ and $\sigma_k^{n+1} = \sigma_k$ for fixed c, m, σ_k not depending on n . Also $t_{n+1}^k \rightarrow^\infty t_{n+2}^k$ for $k = 1, \dots, m$ and $n \in \mathbb{N}$. Because $\xi[\llbracket \nu \rrbracket_v^n / A]$ and ξ'_n are identical on $\{A, B_1, \dots, B_l\}$, by Lemma 4.6 we have $t_{n+1}^k \in \llbracket \sigma_k \rrbracket_{\xi'_n, v}$. Thus $\{t_{n+1}^k\}_{n \in \mathbb{N}}$ is a σ_k, Ξ' -sequence. \square

Lemma 5.5. *If $\tau = d^\infty(\bar{\alpha})$ and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence, then $t_n = ct_n^1 \dots t_n^m$ and $\{t_n^k\}_{n \in \mathbb{N}}$ is a σ_k, Ξ' -sequence for each $k = 1, \dots, m$ where $c \in \text{Constr}(d)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_m)$ and $\Xi' = \Xi[\mathcal{T}]$ where $\mathcal{T} = \{A'\}_{A' \in V_T}$ and $\tau_A = \tau$ and $\tau_{B_j} = \alpha_j$ and $\tau_{A'} = A'$ for $A' \notin \{A, B_1, \dots, B_l\}$ and B_1, \dots, B_l are the parameter type variables of d and A is the recursive type variable of d .*

Proof. The proof is analogous to the proof of Lemma 5.4, but using the fact that $\llbracket d \rrbracket_{\xi, v}^\infty = \Phi_{d, \xi, v}(\llbracket d \rrbracket_{\xi, v}^\infty)$. \square

Definition 5.6. Let S^ν be the set of triples $(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}})$ such that τ is strictly positive, $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is ν -hereditary, and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence. By corecursion we define a function $f^\nu : S^\nu \rightarrow \mathbb{T}^\infty$. Let $\{t_n\}_{n \in \mathbb{N}}$ be a τ, Ξ -sequence. First note that if $\tau = A$ then we may assume $\Xi = \Xi'$, because as long as $\tau = A$ and $\Xi = \Xi'[\mathcal{T}]$, the sequence $\{t_n\}_{n \in \mathbb{N}}$ is also a τ_A, Ξ' -sequence, so we may use the definition for the case $\tau = \tau_A$ and $\Xi = \Xi'$.

- If τ is closed then $f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) = t_0$.
- If $\tau = A$ then without loss of generality $\Xi = \Xi'$ and by Lemma 5.4 for $n \in \mathbb{N}$ we have $t_{n+1} = ct_{n+1}^1 \dots t_{n+1}^m$ and $\{t_{n+1}^k\}_{n \in \mathbb{N}}$ is a σ_k, Ξ' -sequence for each $k = 1, \dots, m$. Then define $f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) = cr_1 \dots r_m$ where $r_k = f^\nu(\sigma_k, \Xi', \{t_{n+1}^k\}_{n \in \mathbb{N}})$.
- If $\tau = d^\infty(\bar{\alpha})$ then by Lemma 5.5 we have $t_n = ct_n^1 \dots t_n^m$ and $\{t_n^k\}_{n \in \mathbb{N}}$ is a σ_k, Ξ' -sequence for each $k = 1, \dots, m$. Then define $f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) = cr_1 \dots r_m$ where $r_k = f^\nu(\sigma_k, \Xi', \{t_n^k\}_{n \in \mathbb{N}})$.
- If $\tau = \forall i. \tau'$ then $f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) = t_0$.
- If $\tau = \tau_1 \rightarrow \tau_2$ then $f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) = t_0$.

We usually denote $f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}})$ by t_∞ when τ, Ξ and $\{t_n\}_{n \in \mathbb{N}}$ are clear from the context.

Lemma 5.7. *If Ξ is ν -hereditary and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence then $t_0 \rightarrow^\infty t_\infty$.*

Proof. By Lemma 2.6 it suffices to show $t_0 \rightarrow^{2^\infty} t_\infty$. We proceed by coinduction. By the definition of t_∞ there are the following possibilities.

- If τ is closed then $t_\infty = t_0$ so $t_0 \rightarrow^{2^\infty} t_\infty$.
- If $\tau = A$ then without loss of generality $\Xi = \Xi'$ and for $n \in \mathbb{N}$ we have $t_{n+1} = ct_{n+1}^1 \dots t_{n+1}^m$ and $\{t_{n+1}^k\}_{n \in \mathbb{N}}$ is a σ_k, Ξ' -sequence for $k = 1, \dots, m$. Then $t_\infty = cr_1 \dots r_m$ with $r_k = f^\nu(\sigma_k, \Xi', \{t_{n+1}^k\}_{n \in \mathbb{N}})$. By the coinductive hypothesis $t_1^k \rightarrow^{2^\infty} r_k$. Because $t_0 \rightarrow^\infty ct_1^1 \dots t_1^m$, we have $t_0 \rightarrow^{2^\infty} t_\infty$.

- If $\tau = d^\infty(\vec{\alpha})$ then $t_n = ct_n^1 \dots t_n^m$ and $\{t_n^k\}_{n \in \mathbb{N}}$ is a σ_k, Ξ' -sequence for each $k = 1, \dots, m$. Then $t_\infty = cr_1 \dots r_m$ where $r_k = f^\nu(\sigma_k, \Xi', \{t_n^k\}_{n \in \mathbb{N}})$. By the coinductive hypothesis $t_0^k \rightarrow^{2^\infty} r_k$, so $t_0 \rightarrow^{2^\infty} t_\infty$.
- If $\tau = \forall i. \tau^i$ or $\tau = \tau_1 \rightarrow \tau_2$ then $t_\infty = t_0$, so $t_0 \rightarrow^{2^\infty} t_\infty$. \square

We want to show that if Ξ is ν -hereditary and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence, then $t_\infty \in \bigcap_{n \in \mathbb{N}} \llbracket \tau \rrbracket_{\xi_n, v}$ (Corollary 5.19). Together with the above lemma and some auxiliary results this will imply the approximation theorem (Theorem 5.22). First, we need a few more definitions and auxiliary lemmas.

Definition 5.8. Let $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ and $\Xi' = \{\xi'_n\}_{n \in \mathbb{N}}$. We write $\Xi \subseteq \Xi'$ if $\xi_n \subseteq \xi'_n$ for $n \in \mathbb{N}$.

Lemma 5.9. If $\Xi \subseteq \Xi'$ and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence, then $\{t_n\}_{n \in \mathbb{N}}$ is also a τ, Ξ' -sequence.

Proof. Follows from definitions and Lemma 4.8. \square

Lemma 5.10. If $t \rightarrow^\infty t_n \in \llbracket \tau_n \rrbracket_{\xi_n, v}$ and ξ_n is stable for $n \in \mathbb{N}$ then there exists a sequence of terms $\{t'_n\}_{n \in \mathbb{N}}$ such that $t \rightarrow^\infty t'_0$ and $t'_n \in \llbracket \tau_n \rrbracket_{\xi_n, v}$ and $t'_n \rightarrow^\infty t'_{n+1}$ for $n \in \mathbb{N}$.

Proof. By induction we define the terms w_n and t'_n such that $t_n \rightarrow^\infty w_n \sim_{\mathcal{U}} t'_n$ and $\{t'_n\}_{n \in \mathbb{N}}$ satisfies the required properties. See Figure 1. We take $t'_0 = w_0 = t_0$. For the inductive step, assume w_n and t'_n are defined. By Lemma 2.4 and confluence modulo \mathcal{U} there are w_{n+1} and w'_{n+1} such that $t_{n+1} \rightarrow^\infty w_{n+1} \sim_{\mathcal{U}} w'_{n+1}$ and $w_n \rightarrow^\infty w'_{n+1}$. By Lemma 2.11 there is t'_{n+1} with $t'_n \rightarrow^\infty t'_{n+1}$ and $w'_{n+1} \sim_{\mathcal{U}} t'_{n+1}$. By Lemma 2.10 we have $w_{n+1} \sim_{\mathcal{U}} t'_{n+1}$. Because ξ_{n+1} is stable, by Lemma 4.10 so is $\llbracket \tau_{n+1} \rrbracket_{\xi_{n+1}, v}$. Since $t_{n+1} \in \llbracket \tau_{n+1} \rrbracket_{\xi_{n+1}, v}$ and $t_{n+1} \rightarrow^\infty w_{n+1} \sim_{\mathcal{U}} t'_{n+1}$ we obtain $t'_{n+1} \in \llbracket \tau_{n+1} \rrbracket_{\xi_{n+1}, v}$. \square

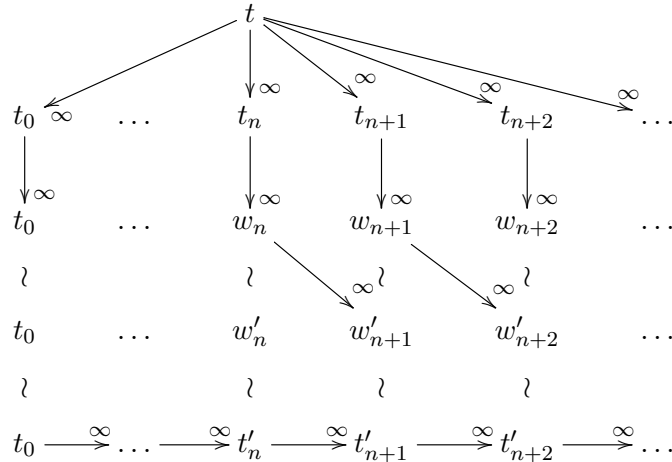


Figure 1: Proof of Lemma 5.10.

Definition 5.11. A ν -hereditary $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is *semi-complete* with Z, ι if $Z \subseteq \Xi$ is stable and for every type variable A and every A, Z -sequence $\{t_n\}_{n \in \mathbb{N}}$ (which is also a A, Ξ -sequence by Lemma 5.9) we have $t_\infty = f^\nu(A, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \iota(A)$. The family Ξ is *complete* if it is semi-complete with Ξ, ξ_m for each $m \in \mathbb{N}$.

Remark 5.12. Note that the definition of “semi-complete” depends on the implicit size variable valuation v , through Ξ and the function f^ν . Let Ξ be ν -hereditary (with v) and semi-complete with Z, ι , with the implicit valuation v . Let i be a fresh size variable. Then by Lemma 5.3 the family Ξ is ν -hereditary with $v[x/i]$ and determined by the same heredity derivation. It is also semi-complete with Z, ι , with the implicit valuation $v[x/i]$. This is because if Ξ is ν -hereditary with $v[x/i]$ and $\{t_n\}_{n \in \mathbb{N}}$ a τ, Ξ -sequence with $v[x/i]$, then it follows from Definition 5.6 and the statements of Lemma 5.4 and Lemma 5.5 that only the type τ , the heredity derivation and the sequence $\{t_n\}_{n \in \mathbb{N}}$ determine the value of $f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}})$. Also note that the property of being an A, Z -sequence does not depend on v , because A is a type variable.

We are now going to show that if $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is complete and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence, then $t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \bigcap_{n \in \mathbb{N}} \llbracket \tau \rrbracket_{\xi_n, v}$ (Corollary 5.14). This is a consequence of the following a bit more general lemma. Its proof is rather long and technical, and therefore delegated to an appendix to make the overall structure of the proof of the approximation theorem clearer.

Lemma 5.13. *If $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is ν -hereditary with v and semi-complete with Z, ι , and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Z -sequence (and thus a τ, Ξ -sequence by Lemma 5.9), then:*

$$t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \llbracket \tau \rrbracket_{\iota, v}.$$

Corollary 5.14. *If $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is complete and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence, then $t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \bigcap_{n \in \mathbb{N}} \llbracket \tau \rrbracket_{\xi_n, v}$.*

We are now going to show that every ν -hereditary family Ξ is complete. To achieve this we show that Ξ^ν is complete (Corollary 5.16), and that if Ξ is complete then so is $\Xi[\mathcal{T}]$ (Lemma 5.17).

Lemma 5.15. *If Ξ is semi-complete with Z, ι then $\Xi[\mathcal{T}]$ is semi-complete with $Z[\mathcal{T}], \iota'$ where $\mathcal{T} = \{\tau_A\}_{A \in V_T}$ and $\iota'(A) = \llbracket \tau_A \rrbracket_{\iota, v}$.*

Proof. Let $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ and $\Xi' = \Xi[\mathcal{T}] = \{\xi'_n\}_{n \in \mathbb{N}}$ and $Z = \{\zeta_n\}_{n \in \mathbb{N}}$ and $Z' = Z[\mathcal{T}] = \{\zeta'_n\}_{n \in \mathbb{N}}$. We have $\zeta'_n(A) = \llbracket \tau_A \rrbracket_{\zeta_n, v} \subseteq \llbracket \tau_A \rrbracket_{\xi_n, v} = \xi'_n(A)$ by Lemma 4.8 because $Z \subseteq \Xi$ and thus $\zeta_n \subseteq \xi_n$. Hence $Z' \subseteq \Xi'$. Let $\{t_n\}_{n \in \mathbb{N}}$ be a A, Z' -sequence, i.e., $t_n \rightarrow^\infty t_{n+1}$ and $t_n \in \llbracket A \rrbracket_{\zeta'_n, v} = \zeta'_n(A) = \llbracket \tau_A \rrbracket_{\zeta_n, v}$ for $n \in \mathbb{N}$. Then $\{t_n\}_{n \in \mathbb{N}}$ is also a τ_A, Z -sequence. Because Ξ is semi-complete with Z, ι , by Lemma 5.13 we have $t_\infty = f^\nu(A, \Xi', \{t_n\}_{n \in \mathbb{N}}) = f^\nu(\tau_A, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \llbracket \tau_A \rrbracket_{\iota, v} = \iota'(A)$. \square

Corollary 5.16. *If Ξ is complete then so is $\Xi[\mathcal{T}]$.*

Lemma 5.17. *Ξ^ν is complete.*

Proof. We show by induction on $m \in \mathbb{N}$ that Ξ^ν is semi-complete with Ξ^ν, ξ_m^ν . We have $\Xi^\nu \subseteq \Xi^\nu$. Also Ξ^ν is stable. Let $\{t_n\}_{n \in \mathbb{N}}$ be a A, Ξ^ν -sequence. We need to show $t_\infty = f^\nu(A, \Xi^\nu, \{t_n\}_{n \in \mathbb{N}}) \in \xi_m^\nu(A) = \llbracket \nu \rrbracket_v^m$. If $m = 0$ then $\llbracket \nu \rrbracket_v^m = \mathbb{T}^\infty$, so $t_\infty \in \xi_m^\nu(A)$. Assume $m = m' + 1$. We have $t_n \in \xi_n^\nu(A) = \llbracket \nu \rrbracket_v^n$. Then by Lemma 5.4 we have $t_{n+1} = ct_{n+1}^1 \dots t_{n+1}^k$ for $n \in \mathbb{N}$, and $\{t_{n+1}^i\}_{n \in \mathbb{N}}$ is a σ_i, Ξ' -sequence for each $i = 1, \dots, k$ where $c \in \text{Constr}(\nu)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$ and $\nu = d_\nu(\vec{\alpha})$ and $\Xi' = \Xi^\nu[\mathcal{T}]$ where $\mathcal{T} = \{\tau_A\}_{A \in V_T}$ and $\tau_{B_j} = \alpha_j$ and $\tau_A = A$ for $A \notin \{B_1, \dots, B_l\}$ and B_1, \dots, B_l are the parameter type variables of d_ν . By the inductive hypothesis Ξ^ν is semi-complete with $\Xi^\nu, \xi_{m'}^\nu$. By Lemma 5.15 we conclude that Ξ' is semi-complete with Ξ', ι where $\iota(A) = \llbracket \tau_A \rrbracket_{\xi_{m'}, v}$, i.e.,

$\iota = \xi_{m'}^\nu(\llbracket \alpha_1 \rrbracket_v / B_1, \dots, \llbracket \alpha_l \rrbracket_v / B_l)$. Because $\{t_{n+1}^i\}_{n \in \mathbb{N}}$ is a σ_i, Ξ' -sequence, by Lemma 5.13 we have $t_\infty^i \in \llbracket \sigma_i \rrbracket_{\iota, v}$. Thus $t_\infty = ct_\infty^1 \dots t_\infty^k \in \Phi_{d_\nu, \iota, v}(\iota(A')) = \Phi_{d_\nu, \iota, v}(\llbracket \nu \rrbracket_v^{m'}) = \llbracket \nu \rrbracket_v^m$, where A' is the recursive type variable of d_ν . Hence $t_\infty \in \xi_{m'}^\nu(A)$. \square

Corollary 5.18. *Every ν -hereditary family is complete.*

Proof. Follows by induction from Lemma 5.17 and Corollary 5.16. \square

Corollary 5.19. *If $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is ν -hereditary and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Ξ -sequence, then $t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \bigcap_{n \in \mathbb{N}} \llbracket \tau \rrbracket_{\xi_n, v}$.*

Proof. Follows from Corollary 5.14 and Corollary 5.18. \square

We are now going to show that $\llbracket \nu \rrbracket_v^\omega = \llbracket \nu \rrbracket_v^\infty$, i.e., ω iterations suffice to reach the fixpoint for any coinductive type. For this we need the following lemma about intersection of valuations. We define $\bigcap_{n \in \mathbb{N}} \xi_n$ by $(\bigcap_{n \in \mathbb{N}} \xi_n)(A) = \bigcap_{n \in \mathbb{N}} \xi_n(A)$ for any A .

Lemma 5.20. *If $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is complete then $\bigcap_{n \in \mathbb{N}} \llbracket \tau \rrbracket_{\xi_n, v} \subseteq \llbracket \tau \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}$ for any strictly positive τ .*

Proof. Induction on τ . The proof is similar to the proof the auxiliary Lemma A.1 in Appendix A. We treat three cases that differ more substantially.

- If $\tau = d_\mu^s(\vec{\alpha})$ then $\bigcap_{n \in \mathbb{N}} \llbracket \tau \rrbracket_{\xi_n, v} = \bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^{\nu(s)}$ where $\mu = d_\mu(\vec{\alpha})$. By induction on \varkappa we show $\bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^\varkappa \subseteq \llbracket \mu \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}^\varkappa$. There are three cases.
 - (1) $\varkappa = 0$. Then $\bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^\varkappa = \bigcap_{n \in \mathbb{N}} \emptyset = \emptyset = \llbracket \mu \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}^\varkappa$.
 - (2) $\varkappa = \varkappa' + 1$. Let $t \in \bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^\varkappa$. Then $t = cu_1 \dots u_k$ with $u_i \in \bigcap_{n \in \mathbb{N}} \llbracket \sigma_i \rrbracket_{\xi_n, v}^{\varkappa'}$ where A is the recursive type variable of d_μ and $c \in \text{Constr}(d_\mu)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$ and B_1, \dots, B_l are the parameter type variables of d_μ and $\Xi' = \Xi[\mathcal{T}]$ and $\Xi' = \{\xi'_n\}_{n \in \mathbb{N}}$ and $\mathcal{T} = \{\tau_A\}_{A \in V_T}$ and $\tau_{B_j} = \alpha_j$ and $\tau_A = \mu^i$ and $\tau_A = A'$ for $A' \notin \{A, B_1, \dots, B_l\}$ where i is a fresh size variable such that $v(i) = \varkappa'$ (by Lemma 4.5 we may assume such a size variable exists). So Ξ' is also complete by Corollary 5.16. By the main inductive hypothesis $u_i \in \llbracket \sigma_i \rrbracket_{\xi, v}^{\varkappa'}$ where $\xi = \bigcap_{n \in \mathbb{N}} \xi'_n$. We have $\xi(A) = \bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^{\varkappa'} \subseteq \llbracket \mu \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}^{\varkappa'}$ by the inductive hypothesis. Also $(\bigcap_{n \in \mathbb{N}} \xi'_n)(B_j) = \bigcap_{n \in \mathbb{N}} \xi'_n(B_j) = \bigcap_{n \in \mathbb{N}} \llbracket \alpha_j \rrbracket_{\xi'_n, v} \subseteq \llbracket \alpha_j \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi'_n, v} = \llbracket \alpha_j \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}$ by the inductive hypothesis and Lemma 4.6, because we may assume $B_1, \dots, B_l \notin \text{TV}(\alpha_j)$. Hence

$$\xi = \left(\bigcap_{n \in \mathbb{N}} \xi_n \right) \left[\begin{array}{l} \llbracket \mu \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}^{\varkappa'} / A, \\ \llbracket \alpha_1 \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v} / B_1, \\ \dots, \\ \llbracket \alpha_l \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v} / B_l \end{array} \right].$$

Therefore $t \in \llbracket \mu \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}^\varkappa$.

- (3) \varkappa is a limit ordinal. Let $t \in \bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^\varkappa = \bigcap_{n \in \mathbb{N}} \bigcup_{\varkappa_n < \varkappa} \llbracket \mu \rrbracket_{\xi_n, v}^{\varkappa_n}$. Then for each $n \in \mathbb{N}$ there is $\varkappa_n < \varkappa$ with $t \in \llbracket \mu \rrbracket_{\xi_n, v}^{\varkappa_n}$, i.e., $t \in \bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^{\varkappa_n}$. We have $\varkappa_n > 0$ is a successor ordinal for $n \in \mathbb{N}$, because $\llbracket \mu \rrbracket_{\xi_n, v}^0 = \emptyset$. Because Ξ is stable by Lemma 5.2, using Lemma A.2 we conclude $t \in \bigcap_{n \in \mathbb{N}} \llbracket \mu \rrbracket_{\xi_n, v}^{\varkappa_0}$. Then $t \in \llbracket \mu \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}^{\varkappa_0} \subseteq \llbracket \mu \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}^\varkappa$ by an argument as in the previous point.

- If $\tau = \forall i.\tau'$ then let $t \in \bigcap_{n \in \mathbb{N}} \llbracket \tau \rrbracket_{\xi_n, v}$. Let $\varkappa \in \Omega$. For $n \in \mathbb{N}$ there is t_n with $t \rightarrow^\infty t_n \in \llbracket \tau' \rrbracket_{\xi_n, v[\varkappa/i]}$. By Lemma 5.2 and Lemma 5.10 there exists a sequence of terms $\{t'_n\}_{n \in \mathbb{N}}$ such that $t \rightarrow^\infty t'_0$ and $t'_n \in \llbracket \tau' \rrbracket_{\xi_n, v[\varkappa/i]}$ and $t'_n \rightarrow^\infty t'_{n+1}$ for $n \in \mathbb{N}$. Thus $\{t'_n\}_{n \in \mathbb{N}}$ is a τ', Ξ -sequence (with $v[\varkappa/i]$). Because Ξ is complete, by Corollary 5.14 there is t^\varkappa with $t \rightarrow^\infty t^\varkappa \in \bigcap_{n \in \mathbb{N}} \llbracket \tau' \rrbracket_{\xi_n, v[\varkappa/i]}$. By the inductive hypothesis $t^\varkappa \in \llbracket \tau' \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v[\varkappa/i]}$. Since $\varkappa \in \Omega$ was arbitrary, this implies $t \in \llbracket \tau \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}$.
- If $\tau = \tau_1 \rightarrow \tau_2$ then let $t \in \bigcap_{n \in \mathbb{N}} \llbracket \tau_1 \rightarrow \tau_2 \rrbracket_{\xi_n, v}$ and $w \in \llbracket \tau_1 \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}$. We have $w \in \bigcap_{n \in \mathbb{N}} \llbracket \tau_1 \rrbracket_{\xi_n, v}$ by Lemma 4.8. Hence there exists a sequence of terms $\{w_n\}_{n \in \mathbb{N}}$ with $tw \rightarrow^\infty w_n \in \llbracket \tau_2 \rrbracket_{\xi_n, v}$. By Lemma 5.2 and Lemma 5.10 there exists a sequence of terms $\{w'_n\}_{n \in \mathbb{N}}$ such that $tw \rightarrow^\infty w'_0$ and $w'_n \in \llbracket \tau_2 \rrbracket_{\xi_n, v}$ and $w'_n \rightarrow^\infty w'_{n+1}$ for $n \in \mathbb{N}$. Thus $\{w'_n\}_{n \in \mathbb{N}}$ is a τ_2, Ξ -sequence. Because Ξ is complete, by Corollary 5.14 we have $w'_\infty \in \bigcap_{n \in \mathbb{N}} \llbracket \tau_2 \rrbracket_{\xi_n, v}$. By the inductive hypothesis $w'_\infty \in \llbracket \tau_2 \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}$. By Lemma 5.7 and Lemma 2.4 we also have $tw \rightarrow^\infty w'_\infty$. This shows $t \in \llbracket \tau \rrbracket_{\bigcap_{n \in \mathbb{N}} \xi_n, v}$. \square

The following lemma shows that for a coinductive type ν we have $\llbracket \nu \rrbracket_v^\omega = \llbracket \nu \rrbracket_v^\infty$. Because we allow only strictly positive coinductive types, ω iterations suffice to reach the fixpoint. A similar result was already obtained in e.g. [1].

Lemma 5.21. $\llbracket \nu \rrbracket_v^\omega = \llbracket \nu \rrbracket_v^\infty$.

Proof. It suffices to show $\llbracket \nu \rrbracket_v^\omega \subseteq \llbracket \nu \rrbracket_v^{\omega+1}$. So let $t \in \llbracket \nu \rrbracket_v^\omega$. Then $t \in \llbracket \nu \rrbracket_v^m$ for each $m \in \mathbb{N}$. So $t = cu_1 \dots u_k$ where $u_i \in \llbracket \sigma_i \rrbracket_{\xi'_m, v}$ for $m \in \mathbb{N}$ where $\Xi' = \{\xi'_m\}_{m \in \mathbb{N}}$ and $\Xi' = \Xi^\nu \llbracket \mathcal{T} \rrbracket$ and $\mathcal{T} = \{\tau_A\}_{A \in V_T}$ and $\tau_{B_j} = \llbracket \alpha_j \rrbracket_v$ and $\tau_A = A$ for $A \notin \{B_1, \dots, B_l\}$ and $\nu = d_\nu(\vec{\alpha})$ and B_1, \dots, B_l are the parameter type variables of d_ν . Note that Ξ' is complete by Corollary 5.18. Hence by Lemma 5.20 we have $u_i \in \llbracket \sigma_i \rrbracket_{\bigcap_{m \in \mathbb{N}} \xi'_m, v}$. Let $\xi' = \bigcap_{m \in \mathbb{N}} \xi'_m$. We have $\xi'(A) = \bigcap_{m \in \mathbb{N}} \xi'_m(A) = \bigcap_{m \in \mathbb{N}} \xi'_m(A) = \bigcap_{m \in \mathbb{N}} \llbracket \nu \rrbracket_v^m = \llbracket \nu \rrbracket_v^\omega$ where A is the recursive type variable of d_ν , and $\xi'(B_j) = \llbracket \alpha_j \rrbracket_v$. Therefore $t \in \llbracket \nu \rrbracket_v^{\omega+1}$. \square

Finally, we prove the approximation theorem. Lemma 5.10, Lemma 5.7, Corollary 5.19 and Lemma 5.21 are used in the proof.

Theorem 5.22 (Approximation Theorem). *If $t \rightarrow^\infty t_n \in \llbracket \nu \rrbracket_v^n$ for $n \in \mathbb{N}$ then there exists $t_\infty \in \llbracket \nu \rrbracket_v^\infty$ such that $t \rightarrow^\infty t_\infty$.*

Proof. By Lemma 5.10 there exists a sequence of terms $\{t_n\}_{n \in \mathbb{N}}$ such that $t \rightarrow^\infty t_0$ and $t_n \in \llbracket \nu \rrbracket_v^n$ and $t_n \rightarrow^\infty t_{n+1}$ for $n \in \mathbb{N}$. Hence $\{t_n\}_{n \in \mathbb{N}}$ is a A, Ξ^ν -sequence. By Lemma 5.7 we have $t_0 \rightarrow^\infty t_\infty$, and hence $t \rightarrow^\infty t_\infty$ by Lemma 2.4. By Corollary 5.19 we have $t_\infty \in \bigcap_{n \in \mathbb{N}} \llbracket A \rrbracket_{\xi_n^\nu, v} = \bigcap_{n \in \mathbb{N}} \xi_n^\nu(A) = \bigcap_{n \in \mathbb{N}} \llbracket \nu \rrbracket_v^n = \llbracket \nu \rrbracket_v^\omega$. Also $\llbracket \nu \rrbracket_v^\omega = \llbracket \nu \rrbracket_v^\infty$ by Lemma 5.21, so $t_\infty \in \llbracket \nu \rrbracket_v^\infty$. \square

We now precisely formulate the result about approximations of infinite objects informally described in the introduction: if for every approximation u_n of size n of an infinite object u the application tu_n reduces to an approximation of an infinite object of the right type, with the result approximations getting larger as n gets larger, then there is a reduction starting from tu which “in the limit” produces an infinite object of the right type. We show that this follows from the approximation theorem.

First, we show that a weak version of this is a direct consequence of Theorem 5.22.

Proposition 5.23. *Let $t \in \mathbb{T}^\infty$ and let $f : \mathbb{N} \rightarrow \mathbb{N}$ be such that $\lim_{n \rightarrow \infty} f(n) = \infty$. Assume that for every $n \in \mathbb{N}$ and every $u_n \in \llbracket \nu_1 \rrbracket^n$ there is w_n with $tu_n \rightarrow^\infty w_n \in \llbracket \nu_2 \rrbracket^{f(n)}$. Then $t \in \llbracket \nu_1 \rightarrow \nu_2 \rrbracket$, i.e., for every $u \in \llbracket \nu_1 \rrbracket$ there is w with $tu \rightarrow^\infty w \in \llbracket \nu_2 \rrbracket$.*

Proof. Let $u \in \llbracket \nu_1 \rrbracket$. Because $\llbracket \nu_1 \rrbracket = \llbracket \nu_1 \rrbracket^\infty \subseteq \llbracket \nu_1 \rrbracket^n$, for each $n \in \mathbb{N}$ there is w_n with $tu \rightarrow^\infty w_n \in \llbracket \nu_2 \rrbracket^{f(n)}$. Because $\lim_{n \rightarrow \infty} f(n) = \infty$, we may choose a strictly increasing subsequence $\{f(n_k)\}_{k \in \mathbb{N}}$ from the sequence $\{f(n)\}_{n \in \mathbb{N}}$. Then $f(n_k) \geq k$ for $k \in \mathbb{N}$. Hence $\llbracket \nu_2 \rrbracket^{f(n_k)} \subseteq \llbracket \nu_2 \rrbracket^k$. This implies that for each $k \in \mathbb{N}$ there is w_{n_k} with $tu \rightarrow^\infty w_{n_k} \in \llbracket \nu_2 \rrbracket^k$. Now by Theorem 5.22 there is w with $tu \rightarrow^\infty w \in \llbracket \nu_2 \rrbracket^\infty$. \square

The above result is, however, a bit unsatisfying in that the valuation approximations $\llbracket \nu_1 \rrbracket^n$ contain too many terms, i.e., they contain all terms which nest at least n constructors of the coinductive type ν_1 . In particular, the infinite object u is an approximation of itself, on which the above proof relies. It would be closer to informal intuition to weaken the hypothesis in Proposition 5.23 by requiring the approximants of size n to nest exactly n constructors of the approximated coinductive type.

Definition 5.24. Let $\perp = (\lambda x.xx)(\lambda x.xx)$. Note that \perp is the only reduct of \perp .

For a coinductive definition d_ν and $n \in \mathbb{N}$ we define the *strict valuation approximation* $\llbracket d_\nu \rrbracket_{\perp, \xi, v}^n \subseteq \mathbb{T}^\infty$ as follows: $\llbracket d_\nu \rrbracket_{\perp, \xi, v}^0 = \{\perp\}$, $\llbracket d_\nu \rrbracket_{\perp, \xi, v}^{n+1} = \Phi_{d_\nu, \xi, v}(\llbracket d_\nu \rrbracket_{\perp, \xi, v}^n)$. We set $\llbracket \nu \rrbracket_{\perp, \xi, v}^n = \llbracket d \rrbracket_{\perp, \xi[\bar{Y}/\bar{B}], v}^n$ where $\nu = d_\nu(\vec{\alpha})$ is a coinductive type, $Y_j = \llbracket \alpha_j \rrbracket_{\xi, v}$, and \bar{B} are the parameter type variables of d_ν .

The relation \succ is defined coinductively.

$$\begin{array}{c} \overline{t \succ \perp} \quad \overline{x \succ x} \quad \overline{c \succ c} \\ \\ \overline{t \succ t'} \quad \overline{t_1 \succ t'_1 \quad t_2 \succ t'_2} \\ \lambda x.t \succ \lambda x.t' \quad t_1 t_2 \succ t'_1 t'_2 \end{array}$$

$$\overline{\overline{t \succ t' \quad t_k \succ t'_k}} \\ \text{case}(t; \{c_k \vec{x} \Rightarrow t_k\}) \succ \text{case}(t'; \{c_k \vec{x} \Rightarrow t'_k\})$$

In other words, $t \succ t'$ if t' is t with some subterms replaced by \perp . If $t \succ t'$, $t \in \llbracket \nu \rrbracket$ and $t' \in \llbracket \nu \rrbracket_{\perp}^n$ then t' is an *approximant* of t of size n .

Lemma 5.25. *If $t \succ t' \rightarrow u'$ then there is u with $t \rightarrow^\equiv u \succ u'$.*

Proof. Induction on $t' \rightarrow u'$. \square

Lemma 5.26. *If $t \succ t' \rightarrow^\infty u'$ then there is u with $t \rightarrow^\infty u \succ u'$.*

Proof. By coinduction, analysing $t' \rightarrow^\infty u'$ and using Lemma 5.25. More precisely, one defines an appropriate function $f : \mathbb{T}^\infty \times \mathbb{T}^\infty \times \mathbb{T}^\infty \rightarrow \mathbb{T}^\infty$ by corecursion and shows $t \rightarrow^\infty f(t, t', u')$ and $f(t, t', u') \succ u'$ by coinduction separately. \square

A set $X \subseteq \mathbb{T}^\infty$ is *approximation expansion closed* if $t' \in X$ and $t \succ t'$ imply $t \in X$.

Lemma 5.27. *Assume $\xi(A)$ is approximation expansion closed for every A . Then $\llbracket \tau \rrbracket_{\xi, v}$ is approximation expansion closed.*

Proof. Induction on τ , using Lemma 5.26 for the cases $\tau = \tau_1 \rightarrow \tau_2$ and $\tau = \forall i.\tau'$. \square

Theorem 5.28. *Let $t \in \mathbb{T}^\infty$ and let $f : \mathbb{N} \rightarrow \mathbb{N}$ be such that $\lim_{n \rightarrow \infty} f(n) = \infty$. Let $u \in \llbracket \nu_1 \rrbracket$. If for every $n \in \mathbb{N}$ and every $u_n \in \llbracket \nu_1 \rrbracket_\perp^n$ with $u \succ u_n$ there is w_n with $tu_n \rightarrow^\infty w_n \in \llbracket \nu_2 \rrbracket^{f(n)}$, then there is w with $tu \rightarrow^\infty w \in \llbracket \nu_2 \rrbracket$.*

Proof. Let $n \in \mathbb{N}$ and let $u_n \in \llbracket \nu_1 \rrbracket_\perp^n$ be such that $u \succ u_n$. There is w_n with $tu_n \rightarrow^\infty w_n \in \llbracket \nu_2 \rrbracket^{f(n)}$. We have $tu \succ tu_n$. By Lemma 5.26 there is v_n with $tu \rightarrow^\infty v_n \succ w_n$. By Lemma 5.27 we have $v_n \in \llbracket \nu_2 \rrbracket^{f(n)}$. Now, because $\lim_{n \rightarrow \infty} f(n) = \infty$, by an argument like the one in the proof of Proposition 5.23, we may conclude that there is w with $tu \rightarrow^\infty w \in \llbracket \nu_2 \rrbracket$. \square

6. THE TYPE SYSTEM λ^\diamond

In this section we define the type system λ^\diamond which provides a syntactic correctness criterion for finite terms decorated with type information. In the next section we use the approximation theorem to prove soundness: if a finite decorated term t has type τ in the system λ^\diamond then its erasure infinitarily reduces to a $t' \in \llbracket \tau \rrbracket$.

Decorated terms are given by:

$$t ::= x \mid c \mid \lambda x : \tau. t \mid tt \mid ts \mid \Lambda i. t \mid \text{case}(t; \{c_k \vec{x} \Rightarrow t_k\}) \mid \text{fix } f : \tau. t \mid \text{cofix }^j f : \tau. t$$

where $x \in \mathcal{V}$, and $c, c_k \in \mathcal{C}$, and τ is a type, and j is a size variable, and s is a size expression.

We define $s_1 \leq s_2$ iff $v(s_1) \leq v(s_2)$ for every size variable valuation v .

The function tgt that gives the *target* of a type is defined as follows:

- $\text{tgt}(A) = A$, $\text{tgt}(\rho^s) = \rho^s$,
- $\text{tgt}(\tau_1 \rightarrow \tau_2) = \text{tgt}(\tau_2)$,
- $\text{tgt}(\forall i. \tau) = \text{tgt}(\tau)$.

By $\text{chgtgt}(\tau, \alpha)$ we denote the type τ with the target exchanged for α . Formally, $\text{chgtgt}(\tau, \alpha)$ is defined inductively:

- $\text{chgtgt}(A, \alpha) = \alpha$, $\text{tgt}(\rho^s, \alpha) = \alpha$,
- $\text{chgtgt}(\tau_1 \rightarrow \tau_2, \alpha) = \tau_1 \rightarrow \text{chgtgt}(\tau_2, \alpha)$,
- $\text{chgtgt}(\forall i. \tau, \alpha) = \forall i. \text{chgtgt}(\tau, \alpha)$.

Note that free size variables in α may be captured as a result of this operation.

A *context* Γ is a finite map from type variables to types. We write $\Gamma, x : \alpha$ to denote the context Γ' such that $\Gamma'(x) = \alpha$ and $\Gamma'(y) = \Gamma(y)$ for $x \neq y$. A *judgement* has the form $\Gamma \vdash t : \alpha$. The rules of the type system λ^\diamond are presented in Figure 2. Figure 3 defines the subtyping relation used in Figure 2. A closed decorated term t is *typable* if $\vdash t : \tau$ for some τ . In Figure 2 *all types are assumed to be closed* (i.e. they don't contain free type variables, but may contain free size variables). In Figure 2 the type variable A denotes the recursive type variable of the (co)inductive definition considered in a given rule, and \vec{B} denote the parameter type variables.

We now briefly explain the typing rules. The rules (ax), (sub), (lam), (app), (inst), (gen) are standard. The rule (con) allows to type constructors of (co)inductive types. It states that if each argument t_k of the constructor c of a (co)inductive type ρ may be assigned an appropriate type with the size of the recursive occurrences of ρ being s , then $ct_1 \dots t_n$ has type ρ^{s+1} . For instance, for the type of lists of natural numbers $\text{List}(\text{Nat})$, the rule (con) says that if $x : \text{Nat}$ and $y : \text{List}^i(\text{Nat})$ then $\text{cons } xy : \text{List}^{i+1}(\text{Nat})$.

$$\begin{array}{c}
\overline{\Gamma, x : \tau \vdash x : \tau} \text{ (ax)} \quad \frac{\Gamma \vdash t : \tau \quad \tau \sqsubseteq \tau'}{\Gamma \vdash t : \tau'} \text{ (sub)} \\
\\
\frac{\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n) \quad \text{Def}(c) = d \quad \rho = d(\vec{\tau})}{\Gamma \vdash t_k : \sigma_k[\rho^s/A][\vec{\tau}/\vec{B}] \text{ for } k = 1, \dots, n} \text{ (con)} \\
\Gamma \vdash ct_1 \dots t_n : \rho^{s+1} \\
\\
\frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash (\lambda x : \alpha.t) : \alpha \rightarrow \beta} \text{ (lam)} \quad \frac{\Gamma \vdash t : \alpha \rightarrow \beta \quad \Gamma \vdash t' : \alpha}{\Gamma \vdash tt' : \beta} \text{ (app)} \\
\\
\frac{\Gamma \vdash t : \forall i.\tau}{\Gamma \vdash ts : \tau[s/i]} \text{ (inst)} \quad \frac{\Gamma \vdash t : \tau \quad i \notin \text{FSV}(\Gamma)}{\Gamma \vdash \Lambda i.t : \forall i.\tau} \text{ (gen)} \\
\\
\frac{\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k}) \quad \delta_k^l = \sigma_k^l[\rho^s/A][\vec{\tau}/\vec{B}] \quad \rho = d(\vec{\tau})}{\Gamma \vdash t : \rho^{s+1} \quad \Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k} \vdash t_k : \tau} \text{ (case)} \\
\Gamma \vdash \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\}) : \tau \\
\\
\frac{\Gamma, f : \forall j_1 \dots j_n.\mu^i \rightarrow \tau \vdash t : \forall j_1 \dots j_n.\mu^{i+1} \rightarrow \tau \quad i \notin \text{FSV}(\Gamma, \mu, \tau, j_1, \dots, j_n)}{\Gamma \vdash (\text{fix } f : \forall j_1 \dots j_n.\mu \rightarrow \tau.t) : \forall j_1 \dots j_n.\mu \rightarrow \tau} \text{ (fix)} \\
\\
\frac{\Gamma, f : \text{chgtgt}(\tau, \nu^{\min(s,j)}) \vdash t : \text{chgtgt}(\tau, \nu^{\min(s,j+1)}) \quad \text{tgt}(\tau) = \nu^s}{\Gamma \vdash (\text{cofix }^j f : \tau.t) : \tau} \text{ (cofix)} \\
j \notin \text{FSV}(\Gamma) \quad j \notin \text{SV}(\tau)
\end{array}$$

Figure 2: Rules of the type system λ^\diamond

$$\begin{array}{c}
\overline{A \sqsubseteq A} \\
\\
\frac{\alpha_k \sqsubseteq \beta_k \quad s \leq s'}{d_\mu^s(\vec{\alpha}) \sqsubseteq d_\mu^{s'}(\vec{\beta})} \quad \frac{\alpha_k \sqsubseteq \beta_k \quad s \geq s'}{d_\nu^s(\vec{\alpha}) \sqsubseteq d_\nu^{s'}(\vec{\beta})} \\
\\
\frac{\tau \sqsubseteq \tau'}{\forall i.\tau \sqsubseteq \forall i.\tau'} \quad \frac{\alpha' \sqsubseteq \alpha \quad \beta \sqsubseteq \beta'}{\alpha \rightarrow \beta \sqsubseteq \alpha' \rightarrow \beta'}
\end{array}$$

Figure 3: Subtyping rules

The (case) rule allows to type case expressions. If the decorated term t that is matched on has a (co)inductive type ρ^{s+1} , and for each $k = 1, \dots, n$ under the assumption that the arguments of the constructor c_k have appropriate types (with the recursive occurrences of ρ having size s) the branch t_k may be given the type τ , then the case expression has type τ .

The (fix) rule allows to type recursive fixpoint definitions. It essentially requires that we may type the body t under the assumption that f already “works” for smaller elements.

The (cofix) rule allows to type corecursive fixpoint definitions. Essentially, it requires that we may type the body t under the assumption that f already produces a smaller

coinductive object, i.e., that if f produces an object defined up to depth j then t produces an object defined up to depth $j + 1$. The size variable j in $\text{cofix}^j f : \tau.t$ may occur in t . Example 6.3 below shows how this may be used.

Definition 6.1. Let $Y = (\lambda x.\lambda f.f(xxf))(\lambda x.\lambda f.f(xxf))$ be the Turing fixpoint combinator. Note that $Yt \rightarrow^* t(Yt)$ for any term t .

The *erasure* $|t|$ of a decorated term t is defined inductively:

- $|x| = x, |c| = c,$
- $|\lambda x : \tau.t| = \lambda x.|t|, |\Lambda i.t| = |t|,$
- $|t_1 t_2| = |t_1| |t_2|, |ts| = |t|,$
- $|\text{case}(t; \{c_k \vec{x} \Rightarrow t_k\})| = \text{case}(|t|; \{c_k \vec{x} \Rightarrow |t_k|\}),$
- $|\text{fix } f : \tau.t| = Y(\lambda f.|t|), |\text{cofix } f : \tau.t| = Y(\lambda f.|t|).$

6.1. Examples. In this section, we give a few examples of typing derivations in the system λ^\diamond . For the sake of readability, we only indicate how to derive the typings. It is straightforward but tedious to translate the examples into the exact formalism of λ^\diamond .

Example 6.2. We reuse the definitions of Nat and Strm from Example 3.4 (see also Example 4.2). Consider the function

$$\mathbf{t1} = \Lambda i.\lambda s : \text{Strm}^{i+1}.\text{case}(s; \{\text{cons } x t \Rightarrow t\})$$

To type $\mathbf{t1}$ we use the (gen), (lam) and (case) rules. Assume $s : \text{Strm}^{i+1}$. To type the match we need to type the branch. Assuming $x : \text{Nat}$ and $t : \text{Strm}^i$ we have $t : \text{Strm}^i$, so the match has type Strm^i by the (case) rule. Hence, by the (lam) and (gen) rules we obtain $\vdash \mathbf{t1} : \forall i.\text{Strm}^{i+1} \rightarrow \text{Strm}^i$.

Similarly, the function

$$\mathbf{hd} = \Lambda i.\lambda s : \text{Strm}^{i+1}.\text{case}(s; \{\text{cons } x t \Rightarrow x\})$$

may be assigned the type $\forall i.\text{Strm}^{i+1} \rightarrow \text{Nat}$.

Example 6.3. We return to the stream processors from Example 3.7 and Example 4.4.

The function \mathbf{run} which runs a stream processor on a stream is defined by:

$$\begin{aligned} \mathbf{run} &= \text{cofix } \mathbf{run} : \text{SP} \rightarrow \text{Strm} \rightarrow \text{Strm}. \\ &\lambda x : \text{SP}.\lambda y : \text{Strm}. \\ &\text{case}(x; \{\mathbf{out } z \Rightarrow \mathbf{runi } z y\}) \end{aligned}$$

where

$$\begin{aligned} \mathbf{runi} &= \text{fix } \mathbf{runi} : \text{SPi}(\text{SP}) \rightarrow \text{Strm} \rightarrow \text{Strm}^{j+1}. \\ &\lambda z : \text{SPi}(\text{SP}).\lambda y : \text{Strm}. \\ &\text{case}(z; \{\mathbf{get } f \Rightarrow \mathbf{runi } (f(\mathbf{hd} \infty y)) (\mathbf{t1} \infty y), \mathbf{put } n x' \Rightarrow n :: \mathbf{run } x' y\}) \end{aligned}$$

Recall that Strm with no decorations is an abbreviation for Strm^∞ .

We have $\vdash \mathbf{run} : \text{SP} \rightarrow \text{Strm} \rightarrow \text{Strm}$. Indeed, to use the (cofix) typing rule assume

$$\mathbf{run} : \text{SP} \rightarrow \text{Strm} \rightarrow \text{Strm}^j$$

and $x : \text{SP}$ and $y : \text{Strm}$.

- To type \mathbf{runi} assume $\mathbf{runi} : \text{SPi}^k(\text{SP}) \rightarrow \text{Strm} \rightarrow \text{Strm}^{j+1}$ and $z : \text{SPi}^{k+1}(\text{SP})$ and $y : \text{Strm}$. We apply the (case) rule to type the match inside \mathbf{runi} . For this purpose we need to type both branches.

- Assuming $f : \text{Nat} \rightarrow \text{SPi}^k(\text{SP})$, we have $\mathbf{runi}(f(\mathbf{hd} \infty y))(\mathbf{tl} \infty y) : \text{Strm}^{j+1}$ by the (inst), (sub) and (app) rules (note that $\infty + 1 \leq \infty$ on size expressions).
- Assuming $n : \text{Nat}$ and $x' : \text{SP}$, we have $\mathbf{run} x' y : \text{Strm}^j$ by the (app) rule. Thus $n :: \mathbf{run} x' y : \text{Strm}^{j+1}$ by (con).

Hence the match has type Strm^{j+1} by (case). Thus

$$\mathbf{runi} : \text{SPi}(\text{SP}) \rightarrow \text{Strm} \rightarrow \text{Strm}^{j+1}$$

by the (fix) typing rule.

- To type the match inside \mathbf{run} we use the (case) rule. Under the assumption $z : \text{SPi}(\text{SP})$ the term $\mathbf{runi} z y$ has type Strm^{j+1} by the (app) rule. Hence the match has type Strm^{j+1} by the (case) rule.

Now using the (lam) rule we conclude that under the assumption

$$\mathbf{run} : \text{SP} \rightarrow \text{Strm} \rightarrow \text{Strm}^j$$

the body of \mathbf{run} may be typed with $\text{SP} \rightarrow \text{Strm} \rightarrow \text{Strm}^{j+1}$. Hence $\mathbf{run} : \text{SP} \rightarrow \text{Strm} \rightarrow \text{Strm}$ by the (cofix) rule.

Remark 6.4. Strictly speaking, it is possible to type non-productive terms in our system. For instance, the term $t = \mathbf{cofix} f : \text{Strm}^0.f$ has type Strm^0 . However, this is not a problem and it agrees with an intuitive interpretation of the type system: if $\vdash t : \text{Strm}^0$ then t should produce at least 0 elements of a stream, which does not put any restrictions on t . One could exclude such terms by requiring that s in ν^s in the (cofix) typing rule should tend to infinity when the sizes of the arguments having coinductive types tend to infinity. We did not see a compelling reason to incorporate this requirement explicitly into the type system.

6.2. Type checking. Type checking in λ^\diamond is decidable and coNP-complete. Each decorated term has a minimal type, and there exists a polynomial algorithm to infer (a compact representation of) the minimal type. Type checking then reduces to deciding the subtyping relation between the minimal type and the type being checked.

The proof of the following theorem and the details of the type checking algorithm may be found in Appendix C. We only briefly mention this theorem as an interesting ancillary result. We move the details to an appendix, because this result has no connection with the infinitary rewriting semantics which is the main theme of this paper.

Theorem 6.5. *Type checking in the system λ^\diamond is coNP-complete. More precisely, given Γ, t, τ the problem of checking whether $\Gamma \vdash t : \tau$ is coNP-complete.*

Despite the theoretically high complexity, we believe that the type checking algorithm is practical. It is based on a polynomial reduction of the type-checking problem to the validity of a set of constraints in quantifier-free Presburger arithmetic. Deciding the validity of the constraints is coNP-complete [8, 22], but in practice may probably be checked using an SMT-solver such as Z3 [12] or CVC4 [6].

7. SOUNDNESS

In this section we show soundness: if $\vdash t : \tau$ then there is $t' \in \llbracket \tau \rrbracket$ with $|t| \rightarrow^\infty t'$. We show that soundness of the (cofix) typing rule follows from the approximation theorem. This is the main result of the present section. The justification of the remaining rules of λ^\diamond is straightforward if a bit tedious.

We first prove a lemma justifying the correctness of the (cofix) typing rule. This lemma follows from the approximation theorem.

Lemma 7.1. *Let $r = Y(\lambda f.t)$ with $\text{tgt}(\tau) = \nu^s$. Let $r_0 = r$ and $r_{n+1} = t[r_n/f]$ for $n \in \mathbb{N}$. Let $\tau' = \text{chtgt}(\tau, \nu^{\min(s,j)})$ where $j \notin \text{SV}(s, \nu, \tau)$. If for every $n \in \mathbb{N}$ there is r'_n with $r_n \rightarrow^\infty r'_n \in \llbracket \tau' \rrbracket_{v[n/j]}$, then there is r' with $r \rightarrow^\infty r' \in \llbracket \tau \rrbracket_v$.*

Proof. Note that $r \rightarrow^\infty r_n$ for $n \in \mathbb{N}$ follows by induction, using Lemma 2.2. Thus also $r \rightarrow^\infty r'_n$ for $n \in \mathbb{N}$ by Lemma 2.4.

Without loss of generality assume $\tau = \forall i_1. \nu_1^{i_1} \rightarrow \forall i_2. \nu_2^{i_2} \rightarrow \nu^s$. Let $\varkappa_1, \varkappa_2 \in \Omega$ and let $u_1 \in \llbracket \nu_1 \rrbracket_{v[\varkappa_1/i_1]}^{\varkappa_1}$ and $u_2 \in \llbracket \nu_2 \rrbracket_{v[\varkappa_1/i_1, \varkappa_2/i_2]}^{\varkappa_2}$. Then because $j \notin \text{SV}(s, \nu, \tau)$, using Lemma 4.5, we conclude that for every $n \in \mathbb{N}$ there is r''_n with $r'_n u_1 u_2 \rightarrow^\infty r''_n \in \llbracket \nu \rrbracket_{v[\varkappa_1/i_1, \varkappa_2/i_2]}^{\min(m,n)}$ where $m = v[\varkappa_1/i_1, \varkappa_2/i_2](s)$. It suffices to find r' with $ru_1 u_2 \rightarrow^\infty r' \in \llbracket \nu \rrbracket_{v[\varkappa_1/i_1, \varkappa_2/i_2]}^m$.

First assume $m < \omega$. Then $r''_m \in \llbracket \nu \rrbracket_{v[\varkappa_1/i_1, \varkappa_2/i_2]}^m$, so we may take $r' = r''_m$, because $ru_1 u_2 \rightarrow^\infty r''_m u_1 u_2 \rightarrow^\infty r''_m$. So assume $m \geq \omega$. Then for every $n \in \mathbb{N}$ we have $r''_n \in \llbracket \nu \rrbracket_{v[\varkappa_1/i_1, \varkappa_2/i_2]}^n$. Since $ru_1 u_2 \rightarrow^\infty r''_n u_1 u_2 \rightarrow^\infty r''_n$ for $n \in \mathbb{N}$, by Lemma 2.4 and Theorem 5.22 there is r' with $ru_1 u_2 \rightarrow^\infty r' \in \llbracket \nu \rrbracket_{v[\varkappa_1/i_1, \varkappa_2/i_2]}^\infty$. \square

The next lemma is needed for the justification of the (sub) subtyping rule.

Lemma 7.2. *If $\tau \sqsubseteq \tau'$ then $\llbracket \tau \rrbracket_v \subseteq \llbracket \tau' \rrbracket_v$.*

Proof. Induction on τ . If $\tau = d_\mu^s(\vec{\alpha}) \sqsubseteq \tau' = d_\mu^{s'}(\vec{\beta})$ then $s \leq s'$ and $\alpha_i \sqsubseteq \beta_i$. We have $\llbracket \tau \rrbracket_v = \llbracket d_\mu(\vec{\alpha}) \rrbracket_v^{v(s)} = \llbracket d_\mu \rrbracket_{\xi, v}^{v(s)}$ where $\xi(B_i) = \llbracket \alpha_i \rrbracket_v$. By the inductive hypothesis $\llbracket \alpha_i \rrbracket_v \subseteq \llbracket \beta_i \rrbracket_v$. Let $\xi'(B_i) = \llbracket \beta_i \rrbracket_v$. Then $\xi \subseteq \xi'$. By Lemma 4.8 we obtain $\llbracket d_\mu \rrbracket_{\xi, v}^{v(s)} \subseteq \llbracket d_\mu \rrbracket_{\xi', v}^{v(s)}$. Also $v(s) \leq v(s')$ and $\llbracket \tau' \rrbracket_v = \llbracket d_\mu \rrbracket_{\xi', v}^{v(s')}$. Thus $\llbracket \tau \rrbracket_v^{v(s)} = \llbracket d_\mu \rrbracket_{\xi, v}^{v(s)} \subseteq \llbracket d_\mu \rrbracket_{\xi', v}^{v(s)} \subseteq \llbracket d_\mu \rrbracket_{\xi', v}^{v(s')} = \llbracket \tau' \rrbracket_v$.

If $\tau = d_\nu^s(\vec{\alpha}) \sqsubseteq \tau' = d_\nu^{s'}(\vec{\beta})$ then the argument is analogous to the previous case.

If $\tau = \forall i. \tau_1 \sqsubseteq \tau' = \forall i. \tau'_1$ then $\tau_1 \sqsubseteq \tau'_1$. Thus by the inductive hypothesis $\llbracket \tau_1 \rrbracket_{v[\varkappa/i]} \sqsubseteq \llbracket \tau'_1 \rrbracket_{v[\varkappa/i]}$ for $\varkappa \in \Omega$. Hence $\llbracket \tau \rrbracket_v \subseteq \llbracket \tau' \rrbracket_v$.

Finally, assume $\tau = \tau_1 \rightarrow \tau_2 \sqsubseteq \tau' = \tau'_1 \rightarrow \tau'_2$. Then $\tau'_1 \sqsubseteq \tau_1$ and $\tau_2 \sqsubseteq \tau'_2$. Let $t \in \llbracket \tau \rrbracket_v$. Then for every $r \in \llbracket \tau_1 \rrbracket_v$ there is t' with $tr \rightarrow^\infty t' \in \llbracket \tau_2 \rrbracket_v$. Let $r \in \llbracket \tau'_1 \rrbracket_v$. Since $\llbracket \tau'_1 \rrbracket_v \subseteq \llbracket \tau_1 \rrbracket_v$ by the inductive hypothesis, there exists t' with $tr \rightarrow^\infty t' \in \llbracket \tau_2 \rrbracket_v$. But $\llbracket \tau_2 \rrbracket_v \subseteq \llbracket \tau'_2 \rrbracket_v$ by the inductive hypothesis. Hence $t \in \llbracket \tau' \rrbracket_v$. \square

Theorem 7.3 (Soundness). *If $\Gamma \vdash t : \tau$ with $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$ then for every size variable valuation $v : \mathcal{V}_S \rightarrow \Omega$ and all $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$ there exists t' such that $|t| [t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty t' \in \llbracket \tau \rrbracket_v$.*

Proof. By induction on the length of the derivation of the typing judgement, using Lemma 7.1 and Lemma 7.2. Lemma 7.1 is needed to justify the (cofix) typing rule. The proof is rather long but straightforward. The details may be found in an appendix. \square

8. CONCLUSIONS

We introduced an infinitary rewriting semantics for strictly positive nested higher-order (co)inductive types. This may be seen as a refinement and generalization of the notion of productivity in term rewriting to a setting with higher-order functions and with data specified by nested higher-order inductive and coinductive definitions. We showed an approximation theorem: $t \rightarrow^\infty t_n \in \llbracket \nu \rrbracket_v^n$ for $n \in \mathbb{N}$ then there exists $t_\infty \in \llbracket \nu \rrbracket_v^\infty$ such that $t \rightarrow^\infty t_\infty$, where ν is a coinductive type.

In the second part of the paper, we defined a type system λ^\diamond combining simple types with nested higher-order (co)inductive types, and using size restrictions similarly to systems with sized types. We showed how to use the approximation theorem to prove soundness: if a finite decorated term t has type τ in the system then its erasure infinitarily reduces to a $t' \in \llbracket \tau \rrbracket$. Together with confluence modulo \mathcal{U} of the infinitary reduction relation and the stability of $\llbracket \tau \rrbracket$, this implies that any finite typable term has a well-defined interpretation in the right type. This provides an operational interpretation of typable terms which takes into account the “limits” of infinite reduction sequences.

In particular, if a decorated term t has in the system λ^\diamond a simple (co)inductive type ρ such that $\llbracket \rho \rrbracket$ contains only normal forms, then the term $|t|$ is infinitarily weakly normalizing. It then follows from [33] that any outermost-fair, possibly infinite but weakly continuous, reduction sequence starting from $|t|$ ends in a normal form. Intuitively, this means that any “fair” reduction strategy always produces a normal form “in the limit”. For instance, Strm mentioned in the introduction is such a type, i.e., all terms in $\llbracket \text{Strm} \rrbracket$ are normal forms. If all elements of $\llbracket \rho \rrbracket$ are additionally finite, as e.g. with $\rho = \text{Nat}$, then $|t|$ is in fact finitarily weakly normalizing.

We have not shown infinitary weak normalization for terms having function types. Nonetheless, our interpretation of $t \in \llbracket \tau_1 \rightarrow \tau_2 \rrbracket$ is very natural and ensures the productivity of t regarded as a function: we require that for $u \in \llbracket \tau_1 \rrbracket$ there is $u' \in \llbracket \tau_2 \rrbracket$ with $tu \rightarrow^\infty u'$.

In general, it seems desirable to strengthen our rewriting semantics so as to require *all* maximal (in some sense) infinitary reduction sequences to yield a term of the right type, not just the existence of such a reduction. Or one would want to prove strong infinitary normalization for erasures of typable terms. This, however, does not seem easy to establish at present.

Our proof of the approximation theorem is classical. We do not expect any significant problems to arise in an attempt to constructivise our development, but we did not pay enough attention to constructivity issues to claim this with complete certainty.

REFERENCES

- [1] A. Abel. Termination and productivity checking with continuous types. In *TLCA 2003*, pages 1–15, 2003.
- [2] A. Abel. *A Polymorphic Lambda-Calculus with Sized Higher-Order Types*. PhD thesis, Ludwig-Maximilians-Universität München, 2006.
- [3] A. Abel and B. Pientka. Wellfounded recursion with copatterns: a unified approach to termination and productivity. In *ICFP 2013*, pages 185–196, 2013.
- [4] A. Abel and B. Pientka. Well-founded recursion with copatterns and sized types. *J. Funct. Program.*, 26:e2, 2016.
- [5] H. Barendregt. *The lambda calculus: its syntax and semantics*. North-Holland, 1984.
- [6] C. Barrett, C. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli. CVC4. In *CAV 2011*, pages 171–177, 2011.

- [7] G. Barthe, M.J. Frade, E. Giménez, L. Pinto, and T. Uustalu. Type-based termination of recursive definitions. *Mathematical Structures in Computer Science*, 14(1):97–141, 2004.
- [8] I. Borosh and L. Treybing. Bounds on positive integral solutions of linear Diophantine equations. In *Proceedings of the American Mathematical Society*, volume 55, pages 299–304, 1976.
- [9] T. Coquand. Infinite objects in type theory. In *TYPES'93*, pages 62–78, 1993.
- [10] L. Czajka. Confluence of nearly orthogonal infinitary term rewriting systems. In *RTA 2015*, pages 106–126, 2015.
- [11] L. Czajka. A new coinductive confluence proof for infinitary lambda-calculus. Submitted, 2018.
- [12] L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
- [13] F.-J. de Vries. On undefined and meaningless in lambda definability. In *FSCD 2016*, pages 18:1–18:15, 2016.
- [14] E.W. Dijkstra. On the productivity of recursive definitions. 1980.
- [15] J. Endrullis, C. Grabmayer, and D. Hendriks. Data-oblivious stream productivity. In *LPAR 2008*, volume 5530 of *LNCS*, pages 79–96. Springer, 2008.
- [16] J. Endrullis, C. Grabmayer, D. Hendriks, A. Ishihara, and J.W. Klop. Productivity of stream definitions. *Theoretical Computer Science*, 411(4-5):765–782, 2010.
- [17] J. Endrullis, H.H. Hansen, D. Hendriks, A. Polonsky, and A. Silva. A coinductive framework for infinitary rewriting and equational reasoning. In *RTA 2015*, 2015.
- [18] J. Endrullis, H.H. Hansen, D. Hendriks, A. Polonsky, and A. Silva. Coinductive foundations of infinitary rewriting and infinitary equational logic. *Logical Methods in Computer Science*, 14(1), 2018.
- [19] J. Endrullis and D. Hendriks. Lazy productivity via termination. *Theoretical Computer Science*, 412(28):3203–3225, 2011.
- [20] J. Endrullis and A. Polonsky. Infinitary rewriting coinductively. In *TYPES 2011*, pages 16–27, 2011.
- [21] E. Giménez. Codifying guarded definitions with recursive schemes. In *TYPES'94*, pages 39–59, 1994.
- [22] C. Haase. Subclasses of Presburger arithmetic and the weak EXP hierarchy. In *CSL-LICS '14*, pages 47:1–47:10, 2014.
- [23] P. Hancock, D. Pattinson, and N. Ghani. Representations of stream processors using nested fixed points. *Logical Methods in Computer Science*, 5(3:9), 2009.
- [24] J. Hughes, L. Pareto, and A. Sabry. Proving the correctness of reactive systems using sized types. In *POPL'96*, pages 410–423, 1996.
- [25] A. Ishihara. Productivity of algorithmic systems. *SCSS 2008*, pages 81–95, 2008.
- [26] B. Jacobs and J.M.M. Rutten. An introduction to (co)algebras and (co)induction. In *Advanced Topics in Bisimulation and Coinduction*, pages 38–99. Cambridge University Press, 2011.
- [27] R. Kennaway and F.-J. de Vries. Infinitary rewriting. In Terese, editor, *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*, chapter 12, pages 668–711. Cambridge University Press, 2003.
- [28] R. Kennaway, J.W. Klop, M.R. Sleep, and F.-J. de Vries. Infinitary lambda calculi and Böhm models. In *RTA '95*, pages 257–270, 1995.
- [29] R. Kennaway, J.W. Klop, M.R. Sleep, and F.-J. de Vries. Transfinite reductions in orthogonal term rewriting systems. *Information and Computation*, 119(1):18–38, 1995.
- [30] R. Kennaway, J.W. Klop, M.R. Sleep, and F.-J. de Vries. Infinitary lambda calculus. *Theoretical Computer Science*, 175(1):93–125, 1997.
- [31] R. Kennaway, V. van Oostrom, and F.-J. de Vries. Meaningless terms in rewriting. *Journal of Functional and Logic Programming*, 1:1–35, 1999.
- [32] J. Ketema and J.G. Simonsen. Infinitary combinatory reduction systems: Confluence. *Logical Methods in Computer Science*, 5(4), 2009.
- [33] J. Ketema and J.G. Simonsen. Infinitary combinatory reduction systems: Normalising reduction strategies. *Logical Methods in Computer Science*, 6(1), 2010.
- [34] J. Ketema and J.G. Simonsen. Infinitary combinatory reduction systems. *Information and Computation*, 209(6):893–926, 2011.
- [35] D. Kozen and A. Silva. Practical coinduction. *Mathematical Structures in Computer Science*, 27(7):1132–1152, 2017.
- [36] N.R. Krishnaswami and N. Benton. Ultrametric semantics of reactive programs. In *LICS 2011*, pages 257–266, 2011.

- [37] J. Longley. Notions of computability at higher types I. In *Logic Colloquium*, volume 19, pages 32–142, 2000.
- [38] P. Martin-Löf. Mathematics of infinity. In *COLOG-88*, pages 146–197, 1988.
- [39] D. Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2012.
- [40] P. Severi and F.-J. de Vries. Order structures on Böhm-like models. In *CSL 2005*, pages 103–118, 2005.
- [41] P. Severi and F.-J. de Vries. Decomposing the lattice of meaningless sets in the infinitary lambda calculus. In *WoLLIC 2011*, pages 210–227, 2011.
- [42] P. Severi and F.-J. de Vries. Weakening the axiom of overlap in infinitary lambda calculus. In *RTA 2011*, pages 313–328, 2011.
- [43] P. Severi and F.-J. de Vries. Pure type systems with corecursion on streams: from finite to infinitary normalisation. In *ICFP'12*, pages 141–152, 2012.
- [44] B. Sijsma. On the productivity of recursive list definitions. *ACM Trans. Program. Lang. Syst.*, 11(4):633–649, 1989.
- [45] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.
- [46] H. Zantema and M. Raffelsieper. Proving productivity in infinite data structures. In *RTA 2010*, pages 401–416, 2010.

APPENDIX A. PROOFS FOR SECTION 5

This section provides the proof of Lemma 5.13. First, we need two auxiliary lemmas, which are needed only for the proof of Lemma 5.13 (they are not used outside of this appendix).

Lemma A.1. *If τ is strictly positive and ξ_1, ξ_2 are stable then $\llbracket \tau \rrbracket_{\xi_1, v} \cap \llbracket \tau \rrbracket_{\xi_2, v} = \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$, where we define $(\xi_1 \cap \xi_2)(A) = \xi_1(A) \cap \xi_2(A)$ for any type variable A .*

Proof. Induction on τ . Note that it suffices to show $\llbracket \tau \rrbracket_{\xi_1, v} \cap \llbracket \tau \rrbracket_{\xi_2, v} \subseteq \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$, because the inclusion in the other direction follows from Lemma 4.8 (noting that $\xi_1 \cap \xi_2 \subseteq \xi_i$ for $i = 1, 2$).

- If τ is closed then $\llbracket \tau \rrbracket_{\xi_1, v} \cap \llbracket \tau \rrbracket_{\xi_2, v} = \llbracket \tau \rrbracket_{\xi_1, v} \cap \llbracket \tau \rrbracket_{\xi_2, v} = \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$ by Lemma 4.6.
- If $\tau = A$ then $\llbracket \tau \rrbracket_{\xi_1, v} \cap \llbracket \tau \rrbracket_{\xi_2, v} = \xi_1(A) \cap \xi_2(A) = \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$.
- If $\tau = d_\mu^s(\vec{\alpha})$ then $\llbracket \tau \rrbracket_{\xi_1, v} \cap \llbracket \tau \rrbracket_{\xi_2, v} = \llbracket d_\mu \rrbracket_{\xi'_1, v}^{v(s)} \cap \llbracket d_\mu \rrbracket_{\xi'_2, v}^{v(s)}$ where $\xi'_n(B_j) = \llbracket \alpha_j \rrbracket_{\xi_n, v}$ and $\xi'_n(A') = \xi_n(A')$ for $A \notin \{B_1, \dots, B_l\}$ and B_1, \dots, B_l are the parameter type variables of d_μ . By induction on \varkappa we show $\llbracket d_\mu \rrbracket_{\xi'_1, v}^\varkappa \cap \llbracket d_\mu \rrbracket_{\xi'_2, v}^\varkappa \subseteq \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa$. There are three cases.
 - (1) $\varkappa = 0$. Then $\llbracket d_\mu \rrbracket_{\xi'_1, v}^\varkappa \cap \llbracket d_\mu \rrbracket_{\xi'_2, v}^\varkappa = \emptyset \cap \emptyset = \emptyset = \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa$.
 - (2) $\varkappa = \varkappa' + 1$. Let $t \in \llbracket d_\mu \rrbracket_{\xi'_1, v}^\varkappa \cap \llbracket d_\mu \rrbracket_{\xi'_2, v}^\varkappa$. Then $t = cu_1 \dots u_k$ with

$$u_i \in \bigcap_{n \in \{1, 2\}} \llbracket \sigma_i \rrbracket_{\xi'_n \llbracket d_\mu \rrbracket_{\xi'_n, v}^{\varkappa'} / A, v}$$

where A is the recursive type variable of d_μ and $c \in \text{Constr}(d_\mu)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$. By the main inductive hypothesis $u_i \in \llbracket \sigma_i \rrbracket_{\xi, v}$ where

$$\xi = \bigcap_{n \in \{1, 2\}} \xi'_n \llbracket d_\mu \rrbracket_{\xi'_n, v}^{\varkappa'} / A.$$

We have

$$\xi(A) = \bigcap_{n \in \{1, 2\}} \llbracket d_\mu \rrbracket_{\xi'_n, v}^{\varkappa'} = \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^{\varkappa'}$$

by the inductive hypothesis. Hence

$$\xi = (\xi'_1 \cap \xi'_2) \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^{\varkappa'} / A.$$

Therefore $t \in \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa$.

- (3) \varkappa is a limit ordinal. Let

$$t \in \bigcap_{n \in \{1, 2\}} \llbracket d_\mu \rrbracket_{\xi'_n, v}^\varkappa = \bigcap_{n \in \{1, 2\}} \bigcup_{\varkappa_n < \varkappa} \llbracket d_\mu \rrbracket_{\xi'_n, v}^{\varkappa_n}$$

Then for each $n \in \{1, 2\}$ there is $\varkappa_n < \varkappa$ with $t \in \llbracket d_\mu \rrbracket_{\xi'_n, v}^{\varkappa_n}$. We have $\varkappa_n > 0$ is a successor ordinal for $n = 1, 2$, because $\llbracket d_\mu \rrbracket_{\xi'_n, v}^0 = \emptyset$. Thus $t = cu_1 \dots u_k$ with $u_i \in \bigcap_{m \in \{1, 2\}} \llbracket \sigma_i \rrbracket_{\xi'_m \llbracket d_\mu \rrbracket_{\xi'_m, v}^{\varkappa_n - 1} / A, v}$ where A is the recursive type variable of d_μ and $c \in \text{Constr}(d_\mu)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$. By the main inductive hypothesis $u_i \in \llbracket \sigma_i \rrbracket_{\xi, v}$ where $\xi = \bigcap_{m \in \{1, 2\}} \xi'_m \llbracket d_\mu \rrbracket_{\xi'_m, v}^{\varkappa_n - 1} / A$. Without loss of generality assume $\varkappa_1 \leq \varkappa_2$. We have $\xi(A) = \bigcap_{m \in \{1, 2\}} \llbracket d_\mu \rrbracket_{\xi'_m, v}^{\varkappa_n - 1} \subseteq \llbracket d_\mu \rrbracket_{\xi'_1, v}^{\varkappa_2 - 1} \cap \llbracket d_\mu \rrbracket_{\xi'_2, v}^{\varkappa_2 - 1}$. Hence by the inductive hypothesis $\xi(A) \subseteq \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^{\varkappa_2 - 1}$. Thus by Lemma 4.8 we have $u_i \in \llbracket \sigma_i \rrbracket_{\xi', v}$ where $\xi' = (\xi'_1 \cap \xi'_2) \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^{\varkappa_2 - 1} / A$. Therefore $t \in \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^{\varkappa_2} \subseteq \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa$.

We have thus shown $\llbracket \tau \rrbracket_{\xi_{1,v}} \cap \llbracket \tau \rrbracket_{\xi_{2,v}} \subseteq \llbracket d_\mu \rrbracket_{\xi'_1 \cap \xi'_2}^{v(s)}$. Note that ξ'_1, ξ'_2 are stable by Lemma 4.10. We have $(\xi'_1 \cap \xi'_2)(B_j) = \xi'_1(B_j) \cap \xi'_2(B_j) = \llbracket \alpha_j \rrbracket_{\xi'_1, v} \cap \llbracket \alpha_j \rrbracket_{\xi'_2, v} = \llbracket \alpha_j \rrbracket_{\xi'_1 \cap \xi'_2, v} = \llbracket \alpha_j \rrbracket_{\xi_1 \cap \xi_2, v}$ by the inductive hypothesis and Lemma 4.6, because we may assume $B_1, \dots, B_l \notin \text{TV}(\alpha_j)$. Hence $\llbracket \tau \rrbracket_{\xi_{1,v}} \cap \llbracket \tau \rrbracket_{\xi_{2,v}} \subseteq \llbracket d_\mu(\vec{\alpha}) \rrbracket_{\xi_1 \cap \xi_2, v}^{v(s)} = \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$ by Lemma 4.6.

- If $\tau = d_\nu^s(\vec{\alpha})$ then $\llbracket \tau \rrbracket_{\xi_{1,v}} \cap \llbracket \tau \rrbracket_{\xi_{2,v}} = \llbracket d_\nu \rrbracket_{\xi'_1, v}^{v(s)} \cap \llbracket d_\nu \rrbracket_{\xi'_2, v}^{v(s)}$ where $\xi'_n(B_j) = \llbracket \alpha_j \rrbracket_{\xi_n, v}$ and $\xi'_n(A') = \xi_n(A')$ for $A \notin \{B_1, \dots, B_l\}$ and B_1, \dots, B_l are the parameter type variables of d_ν . Note that ξ'_1, ξ'_2 are stable by Lemma 4.10. First, by induction on \varkappa we show $\llbracket d_\nu \rrbracket_{\xi'_1, v}^\varkappa \cap \llbracket d_\nu \rrbracket_{\xi'_2, v}^\varkappa \subseteq \llbracket d_\nu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa$. If $\varkappa = 0$ then $\llbracket d_\nu \rrbracket_{\xi'_1, v}^\varkappa \cap \llbracket d_\nu \rrbracket_{\xi'_2, v}^\varkappa = \mathbb{T}^\infty \cap \mathbb{T}^\infty = \mathbb{T}^\infty = \llbracket d_\nu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa$. So assume $\varkappa = \varkappa' + 1$. Let $t \in \llbracket d_\nu \rrbracket_{\xi'_1, v}^\varkappa \cap \llbracket d_\nu \rrbracket_{\xi'_2, v}^\varkappa$. Then $t = cu_1 \dots u_k$ with $u_i \in \bigcap_{n \in \{1,2\}} \llbracket \sigma_i \rrbracket_{\xi'_n \llbracket \llbracket d_\nu \rrbracket_{\xi'_n, v}^{\varkappa'} / A \rrbracket, v}$ where A is the recursive type variable of d_ν and $c \in \text{Constr}(d_\nu)$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$. By Lemma 4.10 and the main inductive hypothesis $u_i \in \llbracket \sigma_i \rrbracket_{\xi, v}$ where $\xi = \bigcap_{n \in \{1,2\}} \xi'_n \llbracket \llbracket d_\nu \rrbracket_{\xi'_n, v}^{\varkappa'} / A \rrbracket$. By the inductive hypothesis

$$\xi(A) = \llbracket d_\nu \rrbracket_{\xi'_1, v}^{\varkappa'} \cap \llbracket d_\nu \rrbracket_{\xi'_2, v}^{\varkappa'} = \llbracket d_\nu \rrbracket_{\xi'_1 \cap \xi'_2, v}^{\varkappa'}.$$

Hence $t \in \llbracket d_\nu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa$. Finally, assume \varkappa is a limit ordinal. Then

$$\begin{aligned} \bigcap_{n \in \{1,2\}} \llbracket d_\nu \rrbracket_{\xi'_n, v}^\varkappa &= \bigcap_{n \in \{1,2\}} \bigcap_{\varkappa' < \varkappa} \llbracket d_\nu \rrbracket_{\xi'_n, v}^{\varkappa'} \\ &= \bigcap_{\varkappa' < \varkappa} \bigcap_{n \in \{1,2\}} \llbracket d_\nu \rrbracket_{\xi'_n, v}^{\varkappa'} \\ &\subseteq \bigcap_{\varkappa' < \varkappa} \llbracket d_\nu \rrbracket_{\xi'_1 \cap \xi'_2, v}^{\varkappa'} \\ &= \llbracket d_\nu \rrbracket_{\xi'_1 \cap \xi'_2, v}^\varkappa. \end{aligned}$$

Now like in the previous point

$$\begin{aligned} (\xi'_1 \cap \xi'_2)(B_j) &= \xi'_1(B_j) \cap \xi'_2(B_j) \\ &= \llbracket \alpha_j \rrbracket_{\xi'_1, v} \cap \llbracket \alpha_j \rrbracket_{\xi'_2, v} \\ &= \llbracket \alpha_j \rrbracket_{\xi'_1 \cap \xi'_2, v} \\ &= \llbracket \alpha_j \rrbracket_{\xi_1 \cap \xi_2, v} \end{aligned}$$

by the inductive hypothesis and Lemma 4.6. Hence

$$\begin{aligned} \llbracket \tau \rrbracket_{\xi_{1,v}} \cap \llbracket \tau \rrbracket_{\xi_{2,v}} &= \llbracket d_\nu \rrbracket_{\xi'_1, v}^m \cap \llbracket d_\nu \rrbracket_{\xi'_2, v}^m \\ &\subseteq \llbracket d_\nu \rrbracket_{\xi'_1 \cap \xi'_2, v}^m \\ &= \llbracket d_\nu(\vec{\alpha}) \rrbracket_{\xi_1 \cap \xi_2, v}^m \\ &= \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v} \end{aligned}$$

by Lemma 4.6.

- Suppose $\tau = \forall i. \tau'$. Let $t \in \llbracket \tau \rrbracket_{\xi_{1,v}} \cap \llbracket \tau \rrbracket_{\xi_{2,v}}$ and $\varkappa \in \Omega$. There are t_1, t_2 with $t \rightarrow^\infty t_1 \in \llbracket \tau' \rrbracket_{\xi_{1,v}[\varkappa/i]}$ and $t \rightarrow^\infty t_2 \in \llbracket \tau' \rrbracket_{\xi_{2,v}[\varkappa/i]}$. By confluence modulo \mathcal{U} there are t'_1, t'_2 such that $t_1 \rightarrow^\infty t'_1 \sim_{\mathcal{U}} t'_2$ and $t_2 \rightarrow^\infty t'_2$. Using Lemma 4.10 we obtain $t'_2 \in \llbracket \tau' \rrbracket_{\xi_{1,v}[\varkappa/i]} \cap \llbracket \tau' \rrbracket_{\xi_{2,v}[\varkappa/i]}$. Hence $t \rightarrow^\infty t'_2 \in \llbracket \tau' \rrbracket_{\xi_1 \cap \xi_2, v[\varkappa/i]}$ by the inductive hypothesis. Thus $t \in \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$. This shows $\llbracket \tau \rrbracket_{\xi_{1,v}} \cap \llbracket \tau \rrbracket_{\xi_{2,v}} \subseteq \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$.
- Suppose $\tau = \tau_1 \rightarrow \tau_2$. Let $t \in \llbracket \tau_1 \rightarrow \tau_2 \rrbracket_{\xi_{1,v}} \cap \llbracket \tau_1 \rightarrow \tau_2 \rrbracket_{\xi_{2,v}}$. Let $w \in \llbracket \tau_1 \rrbracket_{\xi_1 \cap \xi_2, v}$. We have $w \in \llbracket \tau_1 \rrbracket_{\xi_{1,v}} \cap \llbracket \tau_1 \rrbracket_{\xi_{2,v}}$. Hence there are w_1, w_2 with $tw \rightarrow^\infty w_1 \in \llbracket \tau_2 \rrbracket_{\xi_{1,v}}$ and $tw \rightarrow^\infty w_2 \in \llbracket \tau_2 \rrbracket_{\xi_{2,v}}$. By confluence modulo \mathcal{U} there are w'_1, w'_2 such that $w'_1 \sim_{\mathcal{U}} w'_2$ and $w_i \rightarrow^\infty w'_i$. By Lemma 4.10 both $\llbracket \tau_2 \rrbracket_{\xi_{1,v}}, \llbracket \tau_2 \rrbracket_{\xi_{2,v}}$ are stable, and thus so is $\llbracket \tau_2 \rrbracket_{\xi_1 \cap \xi_2, v}$.

Hence $w'_1 \in \llbracket \tau_2 \rrbracket_{\xi_1, v} \cap \llbracket \tau_2 \rrbracket_{\xi_2, v}$. By the inductive hypothesis $w'_1 \in \llbracket \tau_2 \rrbracket_{\xi_1 \cap \xi_2, v}$. This shows $\llbracket \tau \rrbracket_{\xi_1, v} \cap \llbracket \tau \rrbracket_{\xi_2, v} \subseteq \llbracket \tau \rrbracket_{\xi_1 \cap \xi_2, v}$. \square

Lemma A.2. *If ξ_1, ξ_2 are stable then $\llbracket \mu \rrbracket_{\xi_1, v}^{\varkappa_1} \cap \llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa_2} \subseteq \llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa_1}$.*

Proof. Induction on \varkappa_1 . We may assume $\varkappa_1 < \varkappa_2$. If $\varkappa_1 = 0$ then $\llbracket \mu \rrbracket_{\xi_1, v}^{\varkappa_1} \cap \llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa_2} = \emptyset \cap \llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa_2} = \emptyset = \llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa_1}$.

If \varkappa_1 is a limit ordinal then there exists $\varkappa_0 < \varkappa_1$ with $t \in \llbracket \mu \rrbracket_{\xi, v}^{\varkappa_0}$ and we may use the inductive hypothesis.

If $\varkappa_1 = \varkappa'_1 + 1$ then we may assume $\varkappa_2 = \varkappa'_2 + 1$. Let $t \in \llbracket \mu \rrbracket_{\xi_1, v}^{\varkappa_1} \cap \llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa_2}$. Then $t = ct_1 \dots t_k$ with $t_i \in \llbracket \sigma_i \rrbracket_{\zeta_1, v} \cap \llbracket \sigma_i \rrbracket_{\zeta_2, v}$, $c \in \text{Constr}(\mu)$, $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$ and (using Lemma 4.6)

$$\zeta_l = \xi_l[\llbracket \mu \rrbracket_{\xi_l, v}^{\varkappa'_l} / A, \llbracket \alpha_1 \rrbracket_{\xi_l, v} / B_1, \dots, \llbracket \alpha_k \rrbracket_{\xi_l, v} / B_k]$$

for $l = 1, 2$, and $\mu = d_\mu(\vec{\alpha})$, and B_1, \dots, B_k are the parameter type variables of d_μ , and A is the recursive type variable of d_μ . By Lemma 4.10 the valuations ζ_1, ζ_2 are stable. By Lemma A.1 we have $t_i \in \llbracket \sigma_i \rrbracket_{\zeta_1 \cap \zeta_2, v}$. Using the inductive hypothesis, Lemma A.1 and Lemma 4.8 we conclude that $t_i \in \llbracket \sigma_i \rrbracket_{\zeta, v}$ where

$$\zeta = \xi'[\llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa'_1} / A, \llbracket \alpha_1 \rrbracket_{\xi_1 \cap \xi_2, v} / B_1, \dots, \llbracket \alpha_k \rrbracket_{\xi_1 \cap \xi_2, v} / B_k].$$

By Lemma 4.8 we have $\zeta \subseteq \zeta'$ where

$$\zeta' = \xi_2[\llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa'_1} / A, \llbracket \alpha_1 \rrbracket_{\xi_2, v} / B_1, \dots, \llbracket \alpha_k \rrbracket_{\xi_2, v} / B_k].$$

Hence $t_i \in \llbracket \sigma_i \rrbracket_{\zeta', v}$ by Lemma 4.8. But this by definition implies $t \in \llbracket \mu \rrbracket_{\xi_2, v}^{\varkappa_1}$. \square

Lemma 5.13. *If $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ is ν -hereditary with v and semi-complete with Z, ι , and $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Z -sequence (and thus a τ, Ξ -sequence by Lemma 5.9), then*

$$t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \llbracket \tau \rrbracket_{\iota, v}.$$

Proof. We proceed by induction on τ . So let $Z = \{\zeta_n\}_{n \in \mathbb{N}}$ be stable and let $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ be ν -hereditary with v and semi-complete with Z, ι , and let $\{t_n\}_{n \in \mathbb{N}}$ be a τ, Z -sequence. By the definition of t_∞ there are the following possibilities.

- If τ is closed then $t_\infty = t_0 \in \llbracket \tau \rrbracket_{\xi_0, v} = \llbracket \tau \rrbracket_{\iota, v}$ by Lemma 4.6.
 - If $\tau = A$ then $t_\infty \in \iota(A) = \llbracket \tau \rrbracket_{\iota, v}$ because Ξ is semi-complete with Z, ι .
 - If $\tau = \mu^\infty$ with $\mu = d_\mu(\vec{\alpha})$ then let $\Xi' = \Xi[\mathcal{T}]$ where $\mathcal{T} = \{\tau_{A'}\}_{A' \in V_T}$ with $\tau_A = \tau$, $\tau_{B_j} = \alpha_j$ and $\tau_{A'} = A'$ for $A' \notin \{A, B_1, \dots, B_l\}$, where B_1, \dots, B_l are the parameter type variables of d_μ , and A is the recursive type variable of d_μ . Note that Ξ' is ν -hereditary, because Ξ is. Let $Z' = \{\zeta'_n\}_{n \in \mathbb{N}}$ where $\zeta'_n = \zeta_n[\llbracket \alpha_1 \rrbracket_{\zeta_n, v} / B_1, \dots, \llbracket \alpha_l \rrbracket_{\zeta_n, v} / B_l]$. Note that $Z' \subseteq \Xi'$ follows from Lemma 4.8, because $\zeta_n \subseteq \xi_n$ and thus $\llbracket \alpha_j \rrbracket_{\zeta_n, v} \subseteq \llbracket \alpha_j \rrbracket_{\xi_n, v}$. Also, Z' is stable by Lemma 4.10, because Z is. Let $t'(A') = \llbracket \tau_{A'} \rrbracket_{\iota, v}$ for any A' . We show the following.
- (\star) Let $X = \{\chi_n\}_{n \in \mathbb{N}}$ be such that $\chi_n(A') = \zeta'_n(A')$ for $A' \neq A$. If Ξ' is semi-complete with X, ι' then Ξ' is semi-complete with X', ι' where $X' = \{\chi'_n\}_{n \in \mathbb{N}}$ and

$$\chi'_n = \chi_n[\Phi_{d_\mu, \zeta'_n, v}(\chi_n(A)) / A].$$

First note that because Ξ' is semi-complete with X, ι' we have $X \subseteq \Xi'$, so $\chi_n(A) \subseteq \xi'_n(A) = \llbracket \tau_A \rrbracket_{\xi_n, v} = \llbracket \mu \rrbracket_{\xi_n, v}^\infty = \llbracket d_\mu \rrbracket_{\xi'_n, v}^\infty$ by Lemma 4.6 because $\xi'_n(B_j) = \llbracket \alpha_j \rrbracket_{\xi_n, v}$. Also $\zeta'_n \subseteq \xi'_n$ because $Z' \subseteq \Xi'$. Therefore

$$\begin{aligned} \chi'_n(A) &= \Phi_{d_\mu, \zeta'_n, v}(\chi_n(A)) \\ &\subseteq \Phi_{d_\mu, \xi'_n, v}(\llbracket d_\mu \rrbracket_{\xi'_n, v}^\infty) \\ &= \llbracket d_\mu \rrbracket_{\xi'_n, v}^\infty \\ &= \xi'_n(A) \end{aligned}$$

by Lemma 4.8. Thus $X' \subseteq \Xi'$. Note that X' is stable by the third point in Lemma 4.10. It remains to show that for any A' and any A', X' -sequence $\{w_n\}_{n \in \mathbb{N}}$ we have $w_\infty = f^\nu(A', \Xi', \{w_n\}_{n \in \mathbb{N}}) \in \iota'(A')$. If $A' \neq A$ then $\{w_n\}_{n \in \mathbb{N}}$ is also a A', X -sequence, so $w_\infty \in \iota'(A')$ follows from the fact that Ξ' is semi-complete with X, ι' . If $A' = A$ then $w_n \in \Phi_{d_\mu, \zeta'_n, v}(\chi_n(A))$ for $n \in \mathbb{N}$. Therefore there exists $c \in \text{Constr}(\mu)$ such that $w_n = cw_n^1 \dots w_n^k$ and $w_n^i \rightarrow^\infty w_{n+1}^i$ and $w_n^i \in \llbracket \sigma_i \rrbracket_{\zeta'_n[\chi_n(A)/A], v}$ where $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$. Because $\chi_n(A') = \zeta'_n(A')$ for $A' \neq A$, we have $\zeta'_n[\chi_n(A)/A] = \chi_n$. Hence $w_n^i \in \llbracket \sigma_i \rrbracket_{\chi_n, v}$. Thus $\{w_n^i\}_{n \in \mathbb{N}}$ is a σ_i, X -sequence. Note that σ_i is smaller than τ . Because Ξ' is semi-complete with X, ι' , by the inductive hypothesis we have $w_\infty^i = f^\nu(\sigma_i, \Xi', \{w_n^i\}_{n \in \mathbb{N}}) \in \llbracket \sigma_i \rrbracket_{\iota', v}$. Note that

$$\begin{aligned} w_\infty &= f^\nu(A, \Xi', \{w_n\}_{n \in \mathbb{N}}) \\ &= f^\nu(\tau_A, \Xi, \{w_n\}_{n \in \mathbb{N}}) \\ &= f^\nu(\tau, \Xi, \{w_n\}_{n \in \mathbb{N}}) \\ &= cw_\infty^1 \dots w_\infty^k. \end{aligned}$$

Hence $w_\infty \in \Phi_{d_\mu, \iota', v}(\iota'(A))$. We have $\iota'(A) = \llbracket \tau \rrbracket_{\iota, v} = \llbracket d_\mu \rrbracket_{\iota', v}^\infty$ by Lemma 4.6 because $\iota'(B_j) = \llbracket \alpha_j \rrbracket_{\iota, v}$ for $j = 1, \dots, l$. Hence $w_\infty \in \Phi_{d_\mu, \iota', v}(\llbracket d_\mu \rrbracket_{\iota', v}^\infty) = \llbracket d_\mu \rrbracket_{\iota', v}^\infty = \llbracket \tau \rrbracket_{\iota, v} = \iota'(A)$. We have thus shown (\star) .

Let $Z^\varkappa = \{\zeta_n^\varkappa\}_{n \in \mathbb{N}}$ be such that $\zeta_n^\varkappa = \zeta'_n[\llbracket \mu \rrbracket_{\zeta_n, v}^\varkappa / A]$. Then $Z^\varkappa \subseteq \Xi'$ follows from Lemma 4.8. Also Z^\varkappa is stable by Lemma 4.10, because Z, Z' are. By induction on \varkappa we show that Ξ' is semi-complete with Z^\varkappa, ι' . We distinguish three cases.

- $\varkappa = 0$. Then $\zeta_n^\varkappa = \zeta'_n[\llbracket \mu \rrbracket_{\zeta_n, v}^0 / A] = \zeta'_n[\emptyset / A]$. We show that for every A' and every A', Z^0 -sequence $\{w_n\}_{n \in \mathbb{N}}$ we have $w_\infty = f^\nu(\tau_{A'}, \Xi, \{w_n\}_{n \in \mathbb{N}}) \in \iota'(A') = \llbracket \tau_{A'} \rrbracket_{\iota, v}$. Let $\{w_n\}_{n \in \mathbb{N}}$ be a A', Z^0 -sequence. If $A' \neq A$ then $\zeta_n^0(A') = \zeta'_n(A') = \llbracket \tau_{A'} \rrbracket_{\zeta_n, v}$, so $\{w_n\}_{n \in \mathbb{N}}$ is a $\tau_{A'}, Z$ -sequence. Moreover, $\tau_{A'} = \alpha_j$ or $\tau_{A'} = A'$, so $\tau_{A'}$ has smaller size than τ . Thus by the (main) inductive hypothesis we have $f^\nu(\tau_{A'}, \Xi, \{w_n\}_{n \in \mathbb{N}}) \in \llbracket \tau_{A'} \rrbracket_{\iota, v}$, i.e., $w_\infty \in \iota'(A')$. If $A' = A$ then $\zeta_n^0(A) = \emptyset$, so there does not exist a A, Z^0 -sequence.
- $\varkappa = \varkappa' + 1$. Then

$$\begin{aligned} \zeta_n^\varkappa &= \zeta'_n[\llbracket \mu \rrbracket_{\zeta_n, v}^{\varkappa'+1} / A] \\ &= \zeta'_n[\Phi_{d_\mu, \zeta'_n, v}(\llbracket \mu \rrbracket_{\zeta_n, v}^{\varkappa'}) / A] \\ &= \zeta_n^{\varkappa'}[\Phi_{d_\mu, \zeta'_n, v}(\zeta_n^{\varkappa'}(A)) / A]. \end{aligned}$$

By the inductive hypothesis Ξ' is semi-complete with $Z^{\varkappa'}, \iota'$. Hence Ξ' is semi-complete with Z^\varkappa, ι' by (\star) .

- \varkappa is a limit ordinal. We need to show that for all A' and every A', Z^\varkappa -sequence $\{w_n\}_{n \in \mathbb{N}}$ we have

$$\begin{aligned} w_\infty &= f^\nu(A', \Xi', \{w_n\}_{n \in \mathbb{N}}) \\ &= f^\nu(\tau_{A'}, \Xi, \{w_n\}_{n \in \mathbb{N}}) \in \iota'(A'). \end{aligned}$$

If $A' \neq A$ then the argument is the same as the one used in showing that Ξ' is semi-complete with Z^0, ι' . So assume $A' = A$. Then $w_n \in \zeta_n^\varkappa(A) = \llbracket \mu \rrbracket_{\zeta_n, v}^\varkappa$ for $n \in \mathbb{N}$. Since \varkappa is a limit ordinal, for each $n \in \mathbb{N}$ there is $\varkappa_n < \varkappa$ such that $w_n \in \llbracket \mu \rrbracket_{\zeta_n, v}^{\varkappa_n}$. Because $w_0 \rightarrow^\infty w_n$ for $n \in \mathbb{N}$ and ζ_0, ζ_n are stable, by Lemma A.2 we obtain $w_n \in \llbracket \mu \rrbracket_{\zeta_n, v}^{\varkappa_0}$ for $n \in \mathbb{N}$. So $\{w_n\}_{n \in \mathbb{N}}$ is a A, Z^{\varkappa_0} -sequence. By the inductive hypothesis Ξ' is semi-complete with Z^{\varkappa_0}, ι' , so $w_\infty \in \iota'(A)$.

Now taking $\varkappa = \infty$ we conclude that Ξ' is semi-complete with Z^∞, ι' . Note that $\zeta_n^\infty(A) = \llbracket \tau \rrbracket_{\zeta_n, v}$ for $n \in \mathbb{N}$. Hence $\{t_n\}_{n \in \mathbb{N}}$ is a A, Z^∞ -sequence, because it is a τ, Z -sequence and $\tau_A = \tau$. Therefore $t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \iota'(A) = \llbracket \tau \rrbracket_{\iota, v}$.

- If $\tau = \nu_0^\infty$ with $\nu_0 = d_{\nu_0}(\vec{\alpha})$ then let $\Xi' = \Xi[\mathcal{T}]$ where $\mathcal{T} = \{\tau_{A'}\}_{A' \in V_T}$ with $\tau_A = \tau$, $\tau_{B_j} = \alpha_j$ and $\tau_{A'} = A'$ for $A' \notin \{A, B_1, \dots, B_l\}$, where B_1, \dots, B_l are the parameter type variables of d_{ν_0} , and A is the recursive type variable of d_{ν_0} . Note that Ξ' is ν -hereditary, because Ξ is. Let $Z' = \{\zeta'_n\}_{n \in \mathbb{N}}$ where $\zeta'_n(A') = \llbracket \tau_{A'} \rrbracket_{\zeta_n, v}$ for all A' . Note that $Z' \subseteq \Xi'$ follows from Lemma 4.8, because $\zeta_n \subseteq \xi_n$ and thus $\llbracket \tau_{A'} \rrbracket_{\zeta_n, v} \subseteq \llbracket \tau_{A'} \rrbracket_{\xi_n, v}$. Also Z' is stable by Lemma 4.10, because Z is. Let $\iota'(A') = \llbracket \tau_{A'} \rrbracket_{\iota, v}$ for any A' . We show the following.

- (\star) Let ι_0 be a type variable valuation such that $\iota_0(A') = \iota'(A')$ for $A' \neq A$. If Ξ' is semi-complete with Z', ι_0 then Ξ' is semi-complete with Z', ι_1 where $\iota_1 = \iota_0[\Phi_{d_{\nu_0}, \iota_0, v}(\iota_0(A))]/A$. Since $Z' \subseteq \Xi'$ and Z' is stable, it suffices to show that for every A', Z' -sequence $\{w_n\}_{n \in \mathbb{N}}$ we have $w_\infty = f^\nu(A', \Xi', \{w_n\}_{n \in \mathbb{N}}) \in \iota_1(A')$. So let $\{w_n\}_{n \in \mathbb{N}}$ be a A', Z' -sequence, i.e. $w_n \in \llbracket A' \rrbracket_{\zeta'_n, v} = \zeta'_n(A')$ and $w_n \rightarrow^\infty w_{n+1}$. If $A' \neq A$ then $\iota_1(A') = \iota_0(A')$. Because Ξ' is semi-complete with Z', ι_0 , we have $w_\infty \in \iota_0(A') = \iota_1(A')$. If $A' = A$ then $w_n \in \llbracket \tau \rrbracket_{\zeta_n, v} = \Phi_{d_{\nu_0}, \zeta'_n, v}(\llbracket \tau \rrbracket_{\zeta_n, v})$. Hence $w_n = cw_n^1 \dots w_n^k$ and $w_n^i \in \llbracket \sigma_i \rrbracket_{\zeta'_n, v}$ and $w_n^i \rightarrow^\infty w_{n+1}^i$ where $c \in \text{Constr}(\nu_0)$, $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_k)$. Thus $\{w_n^i\}_{n \in \mathbb{N}}$ is a σ_i, Z' -sequence. Because Ξ' is semi-complete with Z', ι_0 and σ_i is smaller than τ , by the inductive hypothesis $w_\infty^i = f^\nu(\sigma_i, \Xi', \{w_n^i\}_{n \in \mathbb{N}}) \in \llbracket \sigma_i \rrbracket_{\iota_0, v}$. Note that $w_\infty = cw_\infty^1 \dots w_\infty^k$ by Definition 5.6. Hence $w_\infty \in \Phi_{d_{\nu_0}, \iota_0, v}(\iota_0(A)) = \iota_1(A)$ by Corollary 4.7. We have thus shown (\star).

Let $\iota_0 = \iota'[\mathbb{T}^\infty/A]$. We show that Ξ' is semi-complete with Z', ι_0 . We have already shown $Z' \subseteq \Xi'$ and that Z' is stable. So let $\{w_n\}_{n \in \mathbb{N}}$ be a A', Z' -sequence. We show $w_\infty = f^\nu(A', \Xi', \{w_n\}_{n \in \mathbb{N}}) \in \iota_0(A')$. We have $w_n \in \llbracket A' \rrbracket_{\zeta'_n, v} = \llbracket \tau_{A'} \rrbracket_{\zeta_n, v}$. If $A' \neq A$ then $w_n \in \zeta'_n(A') = \llbracket \tau_{A'} \rrbracket_{\zeta_n, v}$, so $\{w_n\}_{n \in \mathbb{N}}$ is a $\tau_{A'}, Z$ -sequence. Since $\tau_{A'}$ is smaller than τ (because $\tau_{A'} = A'$ or $\tau_{A'} = \alpha_j$) and Ξ is semi-complete with Z, ι , by the inductive hypothesis $w_\infty = f^\nu(A', \Xi', \{w_n\}_{n \in \mathbb{N}}) = f^\nu(\tau_{A'}, \Xi, \{w_n\}_{n \in \mathbb{N}}) \in \llbracket \tau_{A'} \rrbracket_{\iota, v} = \iota'(A') = \iota_0(A')$. If $A' = A$ then $\iota_0(A) = \mathbb{T}^\infty$, so $w_\infty \in \iota_0(A)$.

Now let $\iota_\varkappa = \iota'[\llbracket \nu_0 \rrbracket_{\iota, v}^\varkappa/A]$ for an ordinal \varkappa (recall that $\llbracket \nu_0 \rrbracket_{\iota, v}^0 = \mathbb{T}^\infty$). We show by induction on \varkappa that Ξ' is semi-complete with Z', ι_\varkappa . For $\varkappa = 0$ we have shown this in the previous paragraph. If $\varkappa = \varkappa' + 1$ then this follows from (\star) because $\Phi_{d_{\nu_0}, \iota_{\varkappa'}, v}(\iota_{\varkappa'}(A)) = \Phi_{d_{\nu_0}, \iota_{\varkappa'}, v}(\llbracket \nu_0 \rrbracket_{\iota, v}^{\varkappa'}) = \Phi_{d_{\nu_0}, \iota', v}(\llbracket d_{\nu_0} \rrbracket_{\iota', v}^{\varkappa'}) = \llbracket \nu_0 \rrbracket_{\iota, v}^{\varkappa'+1}$ by Corollary 4.7 and Lemma 4.6 (note that $\iota_{\varkappa'}(B_j) = \iota'(B_j) = \llbracket \alpha_j \rrbracket_{\iota, v}$). So let \varkappa be a limit ordinal. We have already shown $Z' \subseteq \Xi'$ and that Z' is stable. So let $\{w_n\}_{n \in \mathbb{N}}$ be a A', Z' -sequence. By the inductive hypothesis Ξ' is semi-complete with $Z', \iota_{\varkappa'}$ for $\varkappa' < \varkappa$. So if $A' = A$ then $w_\infty \in \bigcap_{\varkappa' < \varkappa} \iota_{\varkappa'}(A) = \bigcap_{\varkappa' < \varkappa} \llbracket \nu_0 \rrbracket_{\iota, v}^{\varkappa'} = \llbracket \nu_0 \rrbracket_{\iota, v}^\varkappa = \iota_\varkappa(A)$. If $A' \neq A$ then $w_\infty \in \iota_0(A') = \iota_\varkappa(A')$.

Now because $\{t_n\}_{n \in \mathbb{N}}$ is a τ, Z -sequence and $\tau_A = \tau$, the sequence $\{t_n\}_{n \in \mathbb{N}}$ is also a A, Z' -sequence. Because Ξ' is semi-complete with Z', ι_∞ and

$$f^\nu(A, \Xi', \{t_n\}_{n \in \mathbb{N}}) = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) = t_\infty,$$

by Definition 5.11 we have $t_\infty \in \iota_\infty(A) = \llbracket \nu_0 \rrbracket_{\iota, v}^\infty = \llbracket \tau \rrbracket_{\iota, v}$.

- If $\tau = \forall i. \tau'$ with i fresh then $t_\infty = t_0$. We need to show $t_0 \in \llbracket \tau \rrbracket_{\iota, v}$. Let $\varkappa \in \Omega$. For $n \in \mathbb{N}$ we have $t_0 \rightarrow^\infty t_n \in \llbracket \tau \rrbracket_{\zeta_n, v}$, so for each $n \in \mathbb{N}$ there exists t'_n with $t_0 \rightarrow^\infty t'_n \in \llbracket \tau' \rrbracket_{\zeta_n, v[\varkappa/i]}$. Because Z is stable, by Lemma 5.10 there is a sequence $\{t''_n\}_{n \in \mathbb{N}}$ such that $t_0 \rightarrow^\infty t''_0$ and $t''_n \rightarrow^\infty t''_{n+1}$ and $t''_n \in \llbracket \tau' \rrbracket_{\zeta_n, v[\varkappa/i]}$ for $n \in \mathbb{N}$. Hence $\{t''_n\}_{n \in \mathbb{N}}$ is a τ', Z -sequence (with $v[\varkappa/i]$). The family Ξ is ν -hereditary with $v[\varkappa/i]$ by Lemma 5.3. Because Ξ is also semi-complete with Z, ι , by Remark 5.12 and the inductive hypothesis there is t^\varkappa with $t_0 \rightarrow^\infty t^\varkappa \in \llbracket \tau' \rrbracket_{\iota, v[\varkappa/i]}$. Because $\varkappa \in \Omega$ was arbitrary, this implies $t_0 \in \llbracket \tau \rrbracket_{\iota, v}$.
- If $\tau = \tau_1 \rightarrow \tau_2$ with τ_1 closed and τ_2 strictly positive, then $t_\infty = t_0$. We need to show $t_0 \in \llbracket \tau \rrbracket_{\iota, v}$. For $n \in \mathbb{N}$ we have $t_0 \rightarrow^\infty t_n \in \llbracket \tau \rrbracket_{\zeta_n, v}$. Let $r \in \llbracket \tau_1 \rrbracket_{\iota, v}$. Because τ_1 is closed, by Lemma 4.6 we have $r \in \llbracket \tau_1 \rrbracket_{\zeta_n, v}$ for each $n \in \mathbb{N}$. Hence for each $n \in \mathbb{N}$ there is t'_n with $t_0 r \rightarrow^\infty t_n r \rightarrow^\infty t'_n \in \llbracket \tau_2 \rrbracket_{\zeta_n, v}$. Because Z is stable, by Lemma 5.10 there is a sequence $\{t''_n\}_{n \in \mathbb{N}}$ such that $t_0 r \rightarrow^\infty t''_0$ and $t''_n \rightarrow^\infty t''_{n+1}$ and $t''_n \in \llbracket \tau_2 \rrbracket_{\zeta_n, v}$ for $n \in \mathbb{N}$. Hence $\{t''_n\}_{n \in \mathbb{N}}$ is a τ_2, Z -sequence. By the inductive hypothesis there is $t'_\infty \in \llbracket \tau_2 \rrbracket_{\iota, v}$ with $t''_0 \rightarrow^\infty t'_\infty$. Since $t_0 r \rightarrow^\infty t''_0$, also $t_0 r \rightarrow^\infty t'_\infty$. We have thus shown $t_0 \in \llbracket \tau \rrbracket_{\iota, v}$. \square

APPENDIX B. PROOFS FOR SECTION 7

This section provides the details of the proof of Theorem 7.3. First, we need two auxiliary lemmas, which are needed only for the proof of Theorem 7.3 (they are not used outside of this appendix).

Lemma B.1. $\llbracket \tau[\tau'/A] \rrbracket_{\xi,v} = \llbracket \tau \rrbracket_{\xi[\llbracket \tau' \rrbracket_{\xi,v}/A],v}$.

Proof. Induction on τ . Let $\xi' = \xi[\llbracket \tau' \rrbracket_{\xi,v}/A]$. If $\tau = \rho^s$ and $\rho = d(\vec{\alpha})$ then $\llbracket \tau[\tau'/A] \rrbracket_{\xi,v} = \llbracket d \rrbracket_{\xi[\vec{X}/\vec{B}],v}^{v(s)}$ where $X_j = \llbracket \alpha_j[\tau'/A] \rrbracket_{\xi,v}$ and \vec{B} are the parameter type variables of d . By the inductive hypothesis $X_j = \llbracket \alpha_j \rrbracket_{\xi',v}$. By Lemma 4.6 we have $\llbracket \tau[\tau'/A] \rrbracket_{\xi,v} = \llbracket d \rrbracket_{\xi[\vec{X}/\vec{B}],v}^{v(s)} = \llbracket d \rrbracket_{\xi'[\vec{X}/\vec{B}],v}^{v(s)} = \llbracket \tau \rrbracket_{\xi',v}$.

If $\tau = A$ then $\llbracket \tau[\tau'/A] \rrbracket_{\xi,v} = \llbracket \tau' \rrbracket_{\xi,v} = \llbracket \tau \rrbracket_{\xi',v}$.

If $\tau = \forall i. \tau_1$ then let $t \in \llbracket \tau[\tau'/A] \rrbracket_{\xi,v}$. Let $\varkappa \in \Omega$. There is t' such that $t \rightarrow^\infty t' \in \llbracket \tau_1[\tau'/A] \rrbracket_{\xi,v[\varkappa/i]}$. By the inductive hypothesis $t' \in \llbracket \tau_1 \rrbracket_{\xi[\llbracket \tau' \rrbracket_{\xi,v[\varkappa/i]}/A],v[\varkappa/i]}$. This implies $t \in \llbracket \tau \rrbracket_{\xi[\llbracket \tau' \rrbracket_{\xi,v}/A],v}$, using Lemma 4.5 (we may assume $i \notin \text{FSV}(\tau')$). The other direction is analogous.

If $\tau = \tau_1 \rightarrow \tau_2$ then let $t \in \llbracket \tau[\tau'/A] \rrbracket_{\xi,v}$. Let $r \in \llbracket \tau_1 \rrbracket_{\xi',v}$. Then $r \in \llbracket \tau_1[\tau'/A] \rrbracket_{\xi,v}$ by the inductive hypothesis. Thus $tr \rightarrow^\infty t' \in \llbracket \tau_2[\tau'/A] \rrbracket_{\xi,v}$. By the inductive hypothesis $t' \in \llbracket \tau_2 \rrbracket_{\xi',v}$. The inclusion in the other direction is analogous. \square

Lemma B.2. $\llbracket \tau[s/i] \rrbracket_v = \llbracket \tau \rrbracket_{v[v(s)/i]}$.

Proof. Induction on τ . \square

Theorem 7.3 (Soundness). *If $\Gamma \vdash t : \tau$ with $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$ then for every size variable valuation $v : \mathcal{V}_S \rightarrow \Omega$ and all $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$ there exists t' such that $|t|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty t' \in \llbracket \tau \rrbracket_v$.*

Proof. By induction on the length of the derivation of the typing judgement. We consider the last rule in the derivation.

- (ax) If $\Gamma, x : \tau \vdash x : \tau$ then the claim follows directly from definitions.
- (sub) Assume $x_1 : \tau_1, \dots, x_n : \tau_n \vdash t : \tau'$ because of $x_1 : \tau_1, \dots, x_n : \tau_n \vdash t : \tau$ and $\tau \sqsubseteq \tau'$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$. By the inductive hypothesis there is t' with $|t|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty t' \in \llbracket \tau \rrbracket_v$. By Lemma 7.2 we also have $t' \in \llbracket \tau' \rrbracket_v$.
- (con) Assume $\Gamma \vdash cr_1 \dots r_n : \rho^{s+1}$ because of $\Gamma \vdash r_k : \sigma_k[\rho^s/A][\vec{\alpha}/\vec{B}]$ for $k = 1, \dots, n$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\text{Def}(c) = d$ and $\rho = d(\vec{\alpha})$ and $\Gamma = x_1 : \tau_1, \dots, x_m : \tau_m$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_m \in \llbracket \tau_m \rrbracket_v$. By the inductive hypothesis for $k = 1, \dots, n$ there is r'_k with $|r_k|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty r'_k \in \llbracket \sigma_k[\rho^s/A][\vec{\alpha}/\vec{B}] \rrbracket_v$. By Lemma B.1 and Lemma 4.6 we have $r'_k \in \llbracket \sigma_k \rrbracket_{\xi[\llbracket \rho \rrbracket_v^{v(s)}/A],v}$ where $\xi(B_j) = \llbracket \alpha_j \rrbracket_v$. Hence

$$|cr_1 \dots r_n|[t_1/x_1, \dots, t_m/x_m] \rightarrow^\infty cr'_1 \dots r'_n \in \llbracket \rho \rrbracket_v^{v(s+1)}.$$

- (lam) Assume $\Gamma \vdash (\lambda x : \alpha. t) : \alpha \rightarrow \beta$ because of $\Gamma, x : \alpha \vdash t : \beta$ and $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$. Let $r \in \llbracket \alpha \rrbracket_v$. By the inductive hypothesis there is t' with $|t|[t_1/x_1, \dots, t_n/x_n, r/x] \rightarrow^\infty t' \in \llbracket \beta \rrbracket_v$. Hence

$$|\lambda x : \alpha. t|[t_1/x_1, \dots, t_n/x_n]r \rightarrow^\infty t' \in \llbracket \beta \rrbracket_v.$$

This implies $|\lambda x : \alpha. t|[t_1/x_1, \dots, t_n/x_n] \in \llbracket \alpha \rightarrow \beta \rrbracket_v$.

- (app) Assume $\Gamma \vdash tt' : \beta$ because of $\Gamma \vdash t : \alpha \rightarrow \beta$ and $\Gamma \vdash t' : \alpha$ and $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$. By the inductive hypothesis there are r, r' such that

$$|t|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty r \in \llbracket \alpha \rightarrow \beta \rrbracket_v$$

and

$$|t'|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty r' \in \llbracket \alpha \rrbracket_v.$$

Hence there is r'' with $|tt'|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty rr' \rightarrow^\infty r'' \in \llbracket \beta \rrbracket_v$.

- (inst) Assume $\Gamma \vdash ts : \tau[s/i]$ because of $\Gamma \vdash t : \forall i.\tau$, where $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$. By the inductive hypothesis $|t|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty t' \in \llbracket \forall i.\tau \rrbracket_v$. Hence there is t'' with $t' \rightarrow^\infty t'' \in \llbracket \tau \rrbracket_{v[v(s)/i]}$. So $|t|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty t'' \in \llbracket \tau[s/i] \rrbracket_v$ by Lemma B.2.

- (gen) Assume $\Gamma \vdash \Lambda i.t : \forall i.\tau$ because of $\Gamma \vdash t : \tau$ with $i \notin \text{FSV}(\Gamma)$ and $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$. Let $\varkappa \in \Omega$. Since $i \notin \text{FSV}(\Gamma)$ by Lemma 4.5 we have $t_k \in \llbracket \tau_k \rrbracket_{v[\varkappa/i]}$ for $k = 1, \dots, n$. By the inductive hypothesis there is r_\varkappa with $|\Lambda i.t|[t_1/x_1, \dots, t_n/x_n] = |t|[t_1/x_1, \dots, t_n/x_n] \rightarrow^\infty r_\varkappa \in \llbracket \tau \rrbracket_{v[\varkappa/i]}$. This implies

$$|t|[t_1/x_1, \dots, t_n/x_n] \in \llbracket \forall i.\tau \rrbracket_v.$$

- (case) Assume $\Gamma \vdash \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\}) : \tau$ because of $\Gamma \vdash t : \rho^{s+1}$ and $\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k} \vdash t_k : \tau$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma_k^l[\rho^s/A][\vec{\alpha}/\vec{B}]$ and $\rho = d(\vec{\alpha})$ and $\Gamma = x_1 : \tau_1, \dots, x_m : \tau_m$. Let $r_1 \in \llbracket \tau_1 \rrbracket_v, \dots, r_m \in \llbracket \tau_m \rrbracket_v$. By the inductive hypothesis there is u with $|t|[r_1/x_1, \dots, r_m/x_m] \rightarrow^\infty u \in \llbracket \rho^{s+1} \rrbracket_v = \llbracket \rho \rrbracket_v^{v(s+1)} = \llbracket \rho \rrbracket_v^{v(s)+1}$ (note that we may have $v(s+1) = v(s) = \infty$, but then the last equation still holds because ∞ is the fixpoint ordinal). Hence $u = c_k u_1 \dots u_{n_k}$ where $u_i \in \llbracket \sigma_k^i \rrbracket_{\xi, v}$ and $\xi(B_j) = \llbracket \alpha_j \rrbracket_v$ and $\xi(A) = \llbracket \rho \rrbracket_v^{v(s)}$. Then by Lemma B.1 we have $u_i \in \llbracket \sigma_k^i \rrbracket_{\xi, v} = \llbracket \delta_k^i \rrbracket_v$. Hence by the inductive hypothesis there is w with $|t_k|[r_1/x_1, \dots, r_m/x_m, u_1/x_k^1, \dots, u_{n_k}/x_k^{n_k}] \rightarrow^\infty w \in \llbracket \tau \rrbracket_v$. Note that also (we may assume $x_k^1, \dots, x_k^{n_k} \notin \text{TV}(r_1, \dots, r_m)$):

$$\begin{aligned} & \text{case}(|t|; \{c_k \vec{x}_k \Rightarrow |t_k| \mid k = 1, \dots, n\})[r_1/x_1, \dots, r_m/x_m] \\ & \rightarrow^\infty \text{case}(u; \{c_k \vec{x}_k \Rightarrow |t_k|[r_1/x_1, \dots, r_m/x_m]\}) \\ & \rightarrow |t_k|[r_1/x_1, \dots, r_m/x_m, u_1/x_k^1, \dots, u_{n_k}/x_k^{n_k}] \\ & \rightarrow^\infty w. \end{aligned}$$

- (fix) Assume $\Gamma \vdash (\text{fix } f : \forall j_1 \dots j_m. \mu \rightarrow \tau.t) : \forall j_1 \dots j_m. \mu \rightarrow \tau$ because of

$$\Gamma, f : \forall j_1 \dots j_m. \mu^i \rightarrow \tau \vdash t : \forall j_1 \dots j_m. \mu^{i+1} \rightarrow \tau$$

where $i \notin \text{FSV}(\Gamma, \mu, \tau, j_1, \dots, j_m)$ and $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$. Let $t' = |t|[t_1/x_1, \dots, t_n/x_n]$ and $r = Y(\lambda f.t')$. Note that $r = |\text{fix } f : \forall j_1 \dots j_m. \mu \rightarrow \tau.t|[t_1/x_1, \dots, t_n/x_n]$. By induction on $\varkappa \in \Omega$ we show $r \in \llbracket \forall j_1 \dots j_m. \mu^i \rightarrow \tau \rrbracket_{v[\varkappa/i]}$. Let $\varkappa_1, \dots, \varkappa_m \in \Omega$. We need to show that for every $u \in \llbracket \mu \rrbracket_{v'}^\varkappa$ there is r' with $ru \rightarrow^\infty r' \in \llbracket \tau \rrbracket_{v'}$, where $v' = v[\varkappa_1/j_1, \dots, \varkappa_m/j_m]$ (recall $i \notin \text{FSV}(\mu, \tau)$). There are three cases.

– $\varkappa = 0$. Then $\llbracket \mu \rrbracket_{v'}^\varkappa = \emptyset$.

– $\varkappa = \varkappa' + 1$. By the inductive hypothesis for \varkappa we have $r \in \llbracket \forall j_1 \dots j_m. \mu^i \rightarrow \tau \rrbracket_{v[\varkappa'/i]}$. By the main inductive hypothesis $t'[r/f] \in \llbracket \forall j_1 \dots j_m. \mu^i \rightarrow \tau \rrbracket_{v[\varkappa'/i]}$. Let $u \in \llbracket \mu \rrbracket_{v'}^\varkappa$. Then there is r' with $ru \rightarrow^* t'[r/f]u \rightarrow^\infty r' \in \llbracket \tau \rrbracket_{v'}$.

– \varkappa is a limit ordinal. Let $u \in \llbracket \mu \rrbracket_{v'}^{\varkappa}$. Then $u \in \llbracket \mu \rrbracket_{v'}^{\varkappa'}$ for some $\varkappa' < \varkappa$. By the inductive hypothesis for \varkappa we have $r \in \llbracket \forall j_1 \dots j_m. \mu^i \rightarrow \tau \rrbracket_{v[\varkappa'/i]}$. Hence, there is r' with $ru \rightarrow^\infty r' \in \llbracket \tau \rrbracket_{v'}$.

We have thus shown that $r \in \llbracket \forall j_1 \dots j_m. \mu^i \rightarrow \tau \rrbracket_{v[\varkappa/i]}$ for all $\varkappa \in \Omega$. In particular, this holds for $\varkappa = \infty$, which implies $r \in \llbracket \forall j_1 \dots j_m. \mu \rightarrow \tau \rrbracket_v$ because $i \notin \text{FSV}(\mu, \tau)$.

(cofix) Assume $\Gamma \vdash (\text{cofix}^j f : \tau.t) : \tau$ because $\Gamma, f : \text{chgtgt}(\tau, \nu^{\min(s,j)}) \vdash t : \text{chgtgt}(\tau, \nu^{\min(s,j+1)})$ and $\text{tgt}(\tau) = \nu^s$ and $j \notin \text{FSV}(\Gamma)$ and $j \notin \text{SV}(\tau)$ and $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$. Let $t_1 \in \llbracket \tau_1 \rrbracket_v, \dots, t_n \in \llbracket \tau_n \rrbracket_v$. Let $t' = |t|[t_1/x_1, \dots, t_n/x_n]$ and $r = \mathbf{Y}(\lambda f.t')$. Let $r_0 = r$ and $r_{n+1} = t'[r_n/f]$ for $n \in \mathbb{N}$. Let $\tau' = \text{chgtgt}(\tau, \nu^{\min(s,j)})$ and $\tau'' = \text{chgtgt}(\tau, \nu^{\min(s,j+1)})$.

By induction on n we show that for each n there is r'_n with $r_n \rightarrow^\infty r'_n \in \llbracket \tau' \rrbracket_{v[n/j]}$. For $n = 0$, we have $r_0 = r \in \llbracket \tau' \rrbracket_{v[0/j]}$ directly from definitions and the fact that $j \notin \text{SV}(\tau)$, because $\llbracket \nu^{\min(s,j)} \rrbracket_{v'[0/j]} = \llbracket \nu \rrbracket_{v'[0/j]}^0 = \mathbb{T}^\infty$ for any v' . So assume $r_n \rightarrow^\infty r'_n \in \llbracket \tau' \rrbracket_{v[n/j]}$. Because $\Gamma, f : \tau' \vdash t : \tau''$, by the main inductive hypothesis there is t'' with $t'[r'_n/f] \rightarrow^\infty t'' \in \llbracket \tau'' \rrbracket_{v[n/j]}$. We have $r_{n+1} = t'[r_n/f] \rightarrow^\infty t'[r'_n/f] \rightarrow^\infty t''$. Take $r'_{n+1} = t''$. Because $r'_{n+1} \in \llbracket \tau'' \rrbracket_{v[n/j]}$ and $j \notin \text{SV}(\tau)$, it follows from definitions and Lemma B.2 that $r'_{n+1} \in \llbracket \tau' \rrbracket_{v[n+1/j]}$.

We have thus shown that for each $n \in \mathbb{N}$ there exists r'_n such that

$$r_n \rightarrow^\infty r'_n \in \llbracket \text{chgtgt}(\tau, \nu^{\min(s,j)}) \rrbracket_{v[n/j]}.$$

Now by Lemma 7.1 there is r' with

$$|\text{cofix}^j f : \tau.t|[t_1/x_1, \dots, t_n/x_n] = r \rightarrow^\infty r' \in \llbracket \tau \rrbracket_v. \quad \square$$

APPENDIX C. TYPE CHECKING AND TYPE INFERENCE

In this section we show that type checking in λ^\diamond is decidable and coNP-complete. First, we show that each decorated term has a minimal type. We give an algorithm to infer the minimal type. Type checking then reduces to deciding the subtyping relation between the minimal type and the type being checked.

C.1. Minimal typing. In this section we show that if t is typable in a context Γ , then there exists a minimal type $\mathcal{T}(\Gamma; t)$ such that $\Gamma \vdash t : \mathcal{T}(\Gamma; t)$ and for every type τ with $\Gamma \vdash t : \tau$ we have $\mathcal{T}(\Gamma; t) \sqsubseteq \tau$. To define $\mathcal{T}(\Gamma; t)$ we first need the definitions of the operations \sqcup and \sqcap on types.

Definition C.1. We define $\tau_1 \sqcup \tau_2$ and $\tau_1 \sqcap \tau_2$ inductively.

- $(\alpha \rightarrow \beta) \sqcup (\alpha' \rightarrow \beta') = (\alpha \sqcap \alpha') \rightarrow (\beta \sqcup \beta')$.
- $(\alpha \rightarrow \beta) \sqcap (\alpha' \rightarrow \beta') = (\alpha \sqcup \alpha') \rightarrow (\beta \sqcap \beta')$.
- $(\forall i. \alpha) \sqcup (\forall i. \alpha') = \forall i. \alpha \sqcup \alpha'$.
- $(\forall i. \alpha) \sqcap (\forall i. \alpha') = \forall i. \alpha \sqcap \alpha'$.
- $(d_\mu^s(\vec{\alpha})) \sqcup (d_\mu^{s'}(\vec{\beta})) = d_\mu^{\max(s, s')}(\vec{\gamma})$ where $\gamma_i = \alpha_i \sqcup \beta_i$.
- $d_\mu^s(\vec{\alpha}) \sqcap d_\mu^{s'}(\vec{\beta}) = d_\mu^{\min(s, s')}(\vec{\gamma})$ where $\gamma_i = \alpha_i \sqcap \beta_i$.
- $d_\nu^s(\vec{\alpha}) \sqcup d_\nu^{s'}(\vec{\beta}) = d_\nu^{\min(s, s')}(\vec{\gamma})$ where $\gamma_i = \alpha_i \sqcup \beta_i$.
- $d_\nu^s(\vec{\alpha}) \sqcap d_\nu^{s'}(\vec{\beta}) = d_\nu^{\max(s, s')}(\vec{\gamma})$ where $\gamma_i = \alpha_i \sqcap \beta_i$.

Lemma C.2. $\tau_1 \sqcap \tau_2 \sqsubseteq \tau_i \sqsubseteq \tau_1 \sqcup \tau_2$.

Proof. By induction on τ_i . □

Lemma C.3. If τ is strictly positive and $\tau \sqsubseteq \tau'$ and $\alpha \sqsubseteq \beta$ then $\tau[\alpha/A] \sqsubseteq \tau'[\beta/A]$. Conversely, if $\tau[\alpha/A] \sqsubseteq \gamma$ (resp. $\gamma \sqsubseteq \tau[\alpha/A]$) then $\gamma = \tau'[\beta/A]$ with $\tau \sqsubseteq \tau'$ (resp. $\tau' \sqsubseteq \tau$) and $\alpha \sqsubseteq \beta$ (resp. $\beta \sqsubseteq \alpha$).

Proof. Induction on τ . □

Lemma C.4. If $s_1 \leq s_2$ then $s[s_1/i] \leq s[s_2/i]$ and $s_1[s/i] \leq s_2[s/i]$.

Proof. Induction on the structure of size expressions. □

Definition C.5. A subexpression occurrence s_0 in a size expression s is *superfluous* in s if it does not occur within a subexpression of the form $s_1 + 1$. For a size expression s satisfying $s \geq 1$ we define the size expression \bar{s} as follows. Let s' be obtained from s by replacing with 0 each superfluous occurrence of a size variable. Then by using obvious identities on size expressions ($\max(0, s) = s$, $\max(s_1 + 1, s_2 + 1) = \max(s_1, s_2) + 1$, $\max(\infty, s) = \infty$, etc) transform s' into $\bar{s} + 1$ (note that $s' = 0$ is not possible because $s \geq 1$). For any size expression s we define the size expression \underline{s} as follows. Let s'' be obtained from s by replacing with $i + 1$ each superfluous occurrence of i . Then by using the identities on size expressions transform s'' into $\underline{s} + 1$ (note that $s'' \geq 1$).

For instance, if $s = \max(i + 1, \min(i, j + 1))$ then $\bar{s} = i$ and $\underline{s} = \max(i, \min(i, j))$. If $s = 0$ then $\bar{s} = 0$.

Lemma C.6.

- (1) If $s \geq 1$ then $s \geq \bar{s} + 1$.

(2) $s \leq \underline{s} + 1$.

Proof. Follows from definitions and Lemma C.4. \square

Lemma C.7. *Assume $s_1 \leq s_2$.*

(1) *If $s_1 \geq 1$ then $\overline{s}_1 \leq \overline{s}_2$.*

(2) $\underline{s}_1 \leq \underline{s}_2$.

Proof. Follows from definitions and Lemma C.4. \square

Lemma C.8. *If $s \geq s_0 + 1$ then $\overline{s} \geq s_0$.*

Proof. By the identities

$$\begin{aligned} \max(\min(s_1, s_2), s_3) &= \min(\max(s_1, s_3), \max(s_2, s_3)) \\ \min(\max(s_1, s_2), s_3) &= \max(\min(s_1, s_3), \min(s_2, s_3)) \end{aligned}$$

we may assume that e.g. $s = \min(\max(i + 1, j, \dots), \max(\dots), \dots)$ and $s_0 = \max(\min(k + 2, i + 2, \dots), \dots)$. Then by the equivalences

$$\begin{aligned} \min(s_1, s_2) \geq s_3 &\leftrightarrow s_1 \geq s_3 \wedge s_2 \geq s_3 \\ \max(s_1, s_2) \leq s_3 &\leftrightarrow s_1 \leq s_3 \wedge s_2 \leq s_3 \end{aligned}$$

it suffices to consider the case e.g. $s = \max(s_1, s_2, s_3)$ and $s_0 = \min(s'_1, s'_2)$ with each s_i, s'_i of the form $j + c$ or c , with j a size variable and $c \in \mathbb{N}$. Note that the operations performed to obtain s, s_0 of this form do not affect whether the occurrences of size variables are superfluous or not, i.e., when transforming s to s' of the required form analogous operations may be simultaneously performed on \overline{s} to obtain \overline{s}' . So it suffices to show $\overline{s} \geq s_0$ for s, s_0 of the form as above. Define s'_0 by replacing in s_0 each $i \in \text{SV}(s) \setminus \text{SV}(\overline{s})$ (i.e. each size variable which occurs only superfluously in s) with ∞ , and simplifying using obvious identities. First note that we may assume that some $i \in \text{SV}(s) \setminus \text{SV}(\overline{s})$ occurs in s_0 , because otherwise $\overline{s} \geq s_0$ follows from $s \geq s_0 + 1$ by setting each $i \in \text{SV}(s) \setminus \text{SV}(\overline{s})$ to 0 and simplifying. We have $s'_0 \geq s_0$. Thus it suffices to show $\overline{s} \geq s'_0$. Assume otherwise, i.e., there is a size variable valuation v such that $v(\overline{s}) < v(s'_0)$. Note that the values of $v(\overline{s})$ and $v(s'_0)$ do not depend on $v(i)$ for $i \in \text{SV}(s) \setminus \text{SV}(\overline{s})$. Hence, we may assume $v(i) = v(\overline{s}) + 1$ for $i \in \text{SV}(s) \setminus \text{SV}(\overline{s})$. Then $v(s) = \max(v(i), v(\overline{s} + 1)) = \max(v(\overline{s}) + 1, v(\overline{s} + 1)) = v(\overline{s}) + 1$ where $i \in \text{SV}(s) \setminus \text{SV}(\overline{s})$ (by how \overline{s} is obtained from s). Also $v(s_0) \geq \min(v(i), v(s'_0))$ where $i \in \text{SV}(s) \setminus \text{SV}(\overline{s})$. Hence $v(s_0) \geq v(\overline{s}) + 1$, because $v(\overline{s}) + 1 \leq v(s'_0)$. Thus $v(\overline{s}) + 2 = v(s_0) + 1 \leq v(s) = v(\overline{s}) + 1$. Contradiction. \square

Lemma C.9. *If $s \leq s_0 + 1$ then $\underline{s} \leq s_0$.*

Proof. Analogously to the proof of Lemma C.8, it suffices to consider the case e.g. $s = \min(s_1, s_2, s_3)$ and $s_0 = \max(s'_1, s'_2)$ with each s_i, s'_i of the form $j + c$ or c , with j a size variable and $c \in \mathbb{N}$. Then it suffices to show: if $\min(i, s_1, s_2, \dots) \leq s_0 + 1$ then $\min(i + 1, s_1, s_2, \dots) \leq s_0 + 1$ with s_0 of the form $\max(\dots)$ as above. We may assume $i \notin \text{SV}(s_1, s_2, \dots)$. There are two cases.

- $i \notin \text{SV}(s_0)$. Suppose $v(s_0) + 1 < \min(v(i) + 1, v(s_1), v(s_2), \dots)$. Then $v'(s_0) + 1 = v(s_0) + 1 < \min(v(i) + 1, v(s_1), v(s_2), \dots) = v'(\min(i, s_1, s_2, \dots))$ for $v' = v[v(i) + 1/i]$. Contradiction.
- $s_0 = \max(i + c, s'_1, s'_2, \dots)$. Then $v(s_0) + 1 = \max(v(i) + c, v(s'_1), v(s'_2), \dots) + 1 \geq v(i) + c + 1 \geq v(i) + 1 \geq v(\min(i + 1, s_1, s_2, \dots))$. \square

To save on notation we introduce a dummy \perp type and set $\perp \sqcup \tau = \tau \sqcup \perp = \tau$. The dummy type \perp is not a valid type, it is used only to simplify the presentation of type sums below. We assume that for every parameter type variable B of a (co)inductive definition d there exists a constructor $c \in \text{Constr}(d)$ such that $B \in \text{TV}(\sigma_i)$ for some i , where $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$. In other words, we do not allow parameter type variables which do not occur in any constructor argument types.

Definition C.10. For a context Γ and a term t we inductively define a minimal type $\mathcal{T}(\Gamma; t)$ of t in Γ .

- $\mathcal{T}(\Gamma, x : \tau; x) = \tau$.
- $\mathcal{T}(\Gamma; ct_1 \dots t_n) = \mu^{\max(s_1, \dots, s_n)+1}$ if $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\mu = d_\mu(\tau_1, \dots, \tau_m)$ and $\text{Def}(c) = d_\mu$ and $\mathcal{T}(\Gamma; t_i) = \sigma'_i[d_\mu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m]$ (we take $s_i = 0$ and $\alpha_j^i = \perp$ if $A \notin \text{TV}(\sigma_i)$, and $\beta_j^i = \perp$ if $B_j \notin \text{TV}(\sigma_i)$) and $\sigma'_i \sqsubseteq \sigma_i$ and $\tau_j = \bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$. Note that $\tau_j \neq \perp$ because of our assumption on the occurrences of B_j .
- $\mathcal{T}(\Gamma; ct_1 \dots t_n) = \nu^{\min(s_1, \dots, s_n)+1}$ if $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\nu = d_\nu(\tau_1, \dots, \tau_m)$ and $\text{Def}(c) = d_\nu$ and $\mathcal{T}(\Gamma; t_i) = \sigma'_i[d_\nu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m]$ (we take $s_i = \infty$ and $\alpha_j^i = \perp$ if $A \notin \text{TV}(\sigma_i)$, and $\beta_j^i = \perp$ if $B_j \notin \text{TV}(\sigma_i)$) and $\sigma'_i \sqsubseteq \sigma_i$ and $\tau_j = \bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$.
- $\mathcal{T}(\Gamma; \lambda x : \alpha. t) = \alpha \rightarrow \beta$ if $\mathcal{T}(\Gamma, x : \alpha; t) = \beta$.
- $\mathcal{T}(\Gamma; tt') = \beta$ if $\mathcal{T}(\Gamma; t) = \alpha \rightarrow \beta$ and $\mathcal{T}(\Gamma; t') \sqsubseteq \alpha$.
- $\mathcal{T}(\Gamma; ts) = \tau[s/i]$ if $\mathcal{T}(\Gamma; t) = \forall i. \tau$.
- $\mathcal{T}(\Gamma; \Lambda i. t) = \forall i. \tau$ if $\mathcal{T}(\Gamma; t) = \tau$ and $i \notin \text{FSV}(\Gamma)$.
- $\mathcal{T}(\Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\})) = \tau$ if $\mathcal{T}(\Gamma; t) = \mu^s$ and $\mu = d(\vec{\beta})$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma'_k[\mu^s/A][\vec{\beta}/\vec{B}]$ and $\mathcal{T}(\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k) = \tau_k$ and $\tau = \bigsqcup_{k=1}^n \tau_k$.
- $\mathcal{T}(\Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\})) = \tau$ if $\mathcal{T}(\Gamma; t) = \nu^s$ and $s \geq 1$ and $\nu = d(\vec{\beta})$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma'_k[\nu^s/A][\vec{\beta}/\vec{B}]$ and $\mathcal{T}(\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k) = \tau_k$ and $\tau = \bigsqcup_{k=1}^n \tau_k$.
- $\mathcal{T}(\Gamma; \text{fix } f : \forall j_1 \dots j_m. \mu \rightarrow \tau. t) = \forall j_1 \dots j_m. \mu \rightarrow \tau$ if

$$\mathcal{T}(\Gamma, f : \forall j_1 \dots j_m. \mu^i \rightarrow \tau; t) \sqsubseteq \forall j_1 \dots j_m. \mu^{i+1} \rightarrow \tau$$
 and $i \notin \text{FSV}(\Gamma, \mu, \tau, j_1, \dots, j_m)$.
- $\mathcal{T}(\Gamma; \text{cofix } f : \tau. t) = \tau$ if

$$\mathcal{T}(\Gamma, f : \text{chgtgt}(\tau, \nu^{\min(s,j)}); t) \sqsubseteq \text{chgtgt}(\tau, \nu^{\min(s,j+1)})$$

and $\text{tgt}(\tau) = \nu^s$ and $j \notin \text{FSV}(\Gamma)$ and $j \notin \text{SV}(\tau)$.

In other cases not accounted for by the above points $\mathcal{T}(\Gamma; t)$ is undefined. In particular, if the result of the operation \sqcup is not defined then $\mathcal{T}(\Gamma; t)$ is undefined. Note that if $\mathcal{T}(\Gamma; t)$ is defined then it is uniquely determined.

Lemma C.11. *If $\mathcal{T}(\Gamma; t)$ is defined then $\Gamma \vdash t : \mathcal{T}(\Gamma; t)$.*

Proof. Induction on the definition of $\mathcal{T}(\Gamma; t)$, using Lemma C.2, Lemma C.3 and Lemma C.6. We show a few representative cases in detail.

- $\mathcal{T}(\Gamma, x : \tau; x) = \tau$. Then $\Gamma, x : \tau \vdash x : \tau$.
- $\mathcal{T}(\Gamma; ct_1 \dots t_n) = \mu^{\max(s_1, \dots, s_n)+1}$ where $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\mu = d_\mu(\tau_1, \dots, \tau_m)$ and $\text{Def}(c) = d_\mu$ and $\mathcal{T}(\Gamma; t_i) = \sigma'_i[d_\mu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m]$ and $\sigma'_i \sqsubseteq \sigma_i$ and $\tau_j = \bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$. We have $\Gamma \vdash t_i : \sigma'_i[d_\mu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m]$ by

the inductive hypothesis. By Lemma C.2 we have $\alpha_j^i \sqsubseteq \tau_j$. Hence, by Lemma C.3 and the (sub) typing rule, $\Gamma \vdash t_i : \sigma_i[d_\mu^{\max(s_1, \dots, s_n)}(\tau_1, \dots, \tau_m)/A][\tau_1/B_1, \dots, \tau_m/B_m]$. Thus $\Gamma \vdash ct_1 \dots t_n : \mu^{\max(s_1, \dots, s_n)+1}$ by the (con) typing rule.

- $\mathcal{T}(\Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\})) = \tau$ where $\mathcal{T}(\Gamma; t) = \mu^s$ and $\mu = d(\vec{\beta})$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma_k^l[\mu^s/A][\vec{\beta}/\vec{B}]$ and $\mathcal{T}(\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k) = \tau_k$ and $\tau = \bigsqcup_{k=1}^n \tau_k$. By the inductive hypothesis $\Gamma \vdash t : \mu^s$ and $\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k} \vdash t_k : \tau_k$. Since $s \leq \underline{s} + 1$ by Lemma C.6, we have $\Gamma \vdash t : \mu^{\underline{s}+1}$ by (sub). By Lemma C.2 and (sub) we have $\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k} \vdash t_k : \tau$. Thus $\Gamma \vdash \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\}) : \tau$ by (case). \square

Lemma C.12.

- (1) For any type τ we have $\tau \sqsubseteq \tau$.
- (2) If $\tau_1 \sqsubseteq \tau_2 \sqsubseteq \tau_3$ then $\tau_1 \sqsubseteq \tau_3$.

Proof. By induction. Point (1) is straightforward, using the definition of \sqsubseteq . We show a few representative cases for point (2).

If $\tau_2 = A$ then we must have $\tau_1 = \tau_3 = A$, so $\tau_1 \sqsubseteq \tau_3$. If $\tau_2 = d_\mu^{s_2}(\vec{\beta})$ then $\tau_1 = d_\mu^{s_1}(\vec{\alpha})$ and $\tau_3 = d_\mu^{s_3}(\vec{\gamma})$ and $s_1 \leq s_2 \leq s_3$ and $\alpha_k \sqsubseteq \beta_k \sqsubseteq \gamma_k$. By the inductive hypothesis $\alpha_k \sqsubseteq \gamma_k$. Hence $\tau_1 \sqsubseteq \tau_3$. If $\tau_2 = \alpha_2 \rightarrow \beta_2$ then $\tau_1 = \alpha_1 \rightarrow \beta_1$ and $\tau_3 = \alpha_3 \rightarrow \beta_3$ and $\alpha_3 \sqsubseteq \alpha_2 \sqsubseteq \alpha_1$ and $\beta_1 \sqsubseteq \beta_2 \sqsubseteq \beta_3$. By the inductive hypothesis $\alpha_3 \sqsubseteq \alpha_1$ and $\beta_1 \sqsubseteq \beta_3$. Thus $\tau_1 \sqsubseteq \tau_3$. \square

Lemma C.13.

- (1) If $\tau_1 \sqcup \tau_2 \sqsubseteq \tau$ then $\tau_1 \sqsubseteq \tau$ and $\tau_2 \sqsubseteq \tau$.
- (2) If $\tau \sqsubseteq \tau_1 \sqcap \tau_2$ then $\tau \sqsubseteq \tau_1$ and $\tau \sqsubseteq \tau_2$.

Proof. We show both points simultaneously by induction on τ .

- (1) Assume $\tau_1 = \alpha \rightarrow \beta$, $\tau_2 = \alpha' \rightarrow \beta'$ and $\tau = \gamma_1 \rightarrow \gamma_2$. Then $\tau_1 \sqcup \tau_2 = (\alpha \sqcap \alpha') \rightarrow (\beta \sqcup \beta')$ and thus $\gamma_1 \sqsubseteq \alpha \sqcap \alpha'$ and $\beta \sqcup \beta' \sqsubseteq \gamma_2$. Hence by the inductive hypothesis $\gamma_1 \sqsubseteq \alpha$, $\gamma_1 \sqsubseteq \alpha'$, $\beta \sqsubseteq \gamma_2$ and $\beta' \sqsubseteq \gamma_2$. This implies $\tau_1 \sqsubseteq \tau$ and $\tau_2 \sqsubseteq \tau$.

Assume $\tau_1 = \forall i. \alpha_i$, $\tau_2 = \forall i. \alpha_i$ and $\tau = \forall i. \gamma_i$. Then $\tau_1 \sqcup \tau_2 = \forall i. \alpha_i \sqcup \alpha_i$. Because $\alpha_i \sqcup \alpha_i \sqsubseteq \gamma_i$, by the inductive hypothesis $\alpha_i \sqsubseteq \gamma_i$ and $\alpha_i \sqsubseteq \gamma_i$. Thus $\tau_1 \sqsubseteq \tau$ and $\tau_2 \sqsubseteq \tau$.

Assume $\tau_1 = d_\mu^{s_1}(\vec{\alpha})$ and $\tau_2 = d_\mu^{s_2}(\vec{\beta})$ and $\tau = d_\mu^s(\vec{\gamma})$. Then $\tau_1 \sqcup \tau_2 = d_\mu^{\max(s_1, s_2)}(\vec{\delta})$ with $\delta_i = \alpha_i \sqcup \beta_i \sqsubseteq \gamma_i$ and $\max(s_1, s_2) \leq s$. By the inductive hypothesis $\alpha_i \sqsubseteq \gamma_i$ and $\beta_i \sqsubseteq \gamma_i$. Also $s_1, s_2 \leq \max(s_1, s_2) \leq s$. Hence $\tau_1 \sqsubseteq \tau$ and $\tau_2 \sqsubseteq \tau$.

Assume $\tau_1 = d_\nu^{s_1}(\vec{\alpha})$ and $\tau_2 = d_\nu^{s_2}(\vec{\beta})$ and $\tau = d_\nu^s(\vec{\gamma})$. Then $\tau_1 \sqcup \tau_2 = d_\nu^{\min(s_1, s_2)}(\vec{\delta})$ with $\delta_i = \alpha_i \sqcup \beta_i \sqsubseteq \gamma_i$ and $\min(s_1, s_2) \geq s$. By the inductive hypothesis $\alpha_i \sqsubseteq \gamma_i$ and $\beta_i \sqsubseteq \gamma_i$. Also $s_1, s_2 \geq \min(s_1, s_2) \geq s$. Hence $\tau_1 \sqsubseteq \tau$ and $\tau_2 \sqsubseteq \tau$.

- (2) The proof for the second point is analogous to the first one. \square

Lemma C.14. Assume $\tau_1 \sqsubseteq \tau'_1$ and $\tau_2 \sqsubseteq \tau'_2$. Then:

- (1) $\tau_1 \sqcup \tau_2 \sqsubseteq \tau'_1 \sqcup \tau'_2$,
- (2) $\tau_1 \sqcap \tau_2 \sqsubseteq \tau'_1 \sqcap \tau'_2$.

Proof. We show both points simultaneously by induction on τ_1 .

- (1) If $\tau_1 = \alpha_1 \rightarrow \beta_1$ and $\tau_2 = \alpha_2 \rightarrow \beta_2$ then $\tau'_1 = \alpha'_1 \rightarrow \beta'_1$ and $\tau'_2 = \alpha'_2 \rightarrow \beta'_2$ with $\alpha'_1 \sqsubseteq \alpha_1$, $\alpha'_2 \sqsubseteq \alpha_2$, $\beta_1 \sqsubseteq \beta'_1$ and $\beta_2 \sqsubseteq \beta'_2$. We have $\tau_1 \sqcup \tau_2 = (\alpha_1 \sqcap \alpha_2) \rightarrow (\beta_1 \sqcup \beta_2)$ and $\tau'_1 \sqcup \tau'_2 = (\alpha'_1 \sqcap \alpha'_2) \rightarrow (\beta'_1 \sqcup \beta'_2)$. By the inductive hypothesis $\alpha'_1 \sqcap \alpha'_2 \sqsubseteq \alpha_1 \sqcap \alpha_2$ and $\beta_1 \sqcup \beta_2 \sqsubseteq \beta'_1 \sqcup \beta'_2$. Hence $\tau_1 \sqcup \tau_2 \sqsubseteq \tau'_1 \sqcup \tau'_2$.

If $\tau_1 = \forall i. \alpha$ and $\tau_2 = \forall i. \beta$ then $\tau'_1 = \forall i. \alpha'$ and $\tau'_2 = \forall i. \beta'$ with $\alpha \sqsubseteq \alpha'$ and $\beta \sqsubseteq \beta'$. By the inductive hypothesis $\alpha \sqcup \beta \sqsubseteq \alpha' \sqcup \beta'$. Thus $\tau_1 \sqcup \tau_2 \sqsubseteq \tau'_1 \sqcup \tau'_2$.

If $\tau_1 = d_\mu^{s_1}(\vec{\alpha})$ and $\tau_2 = d_\mu^{s_2}(\vec{\beta})$ then $\tau'_1 = d_\mu^{s'_1}(\vec{\alpha}')$ and $\tau'_2 = d_\mu^{s'_2}(\vec{\beta}')$ with $\alpha_i \sqsubseteq \alpha'_i$ and $\beta_i \sqsubseteq \beta'_i$ and $s_1 \leq s'_1$ and $s_2 \leq s'_2$. We have $\tau_1 \sqcup \tau_2 = d_\mu^{\max(s_1, s_2)}(\vec{\gamma})$ and $\tau'_1 \sqcup \tau'_2 = d_\mu^{\max(s'_1, s'_2)}(\vec{\gamma}')$, where $\gamma_i = \alpha_i \sqcup \beta_i$ and $\gamma'_i = \alpha'_i \sqcup \beta'_i$. By the inductive hypothesis $\gamma_i \sqsubseteq \gamma'_i$. Also $\max(s_1, s_2) \leq \max(s'_1, s'_2)$. Hence $\tau_1 \sqcup \tau_2 \sqsubseteq \tau'_1 \sqcup \tau'_2$.

If $\tau_1 = d_\nu^{s_1}(\vec{\alpha})$ and $\tau_2 = d_\nu^{s_2}(\vec{\beta})$ then $\tau'_1 = d_\nu^{s'_1}(\vec{\alpha}')$ and $\tau'_2 = d_\nu^{s'_2}(\vec{\beta}')$ with $\alpha_i \sqsubseteq \alpha'_i$ and $\beta_i \sqsubseteq \beta'_i$ and $s_1 \geq s'_1$ and $s_2 \geq s'_2$. We have $\tau_1 \sqcup \tau_2 = d_\nu^{\min(s_1, s_2)}(\vec{\gamma})$ and $\tau'_1 \sqcup \tau'_2 = d_\nu^{\min(s'_1, s'_2)}(\vec{\gamma}')$, where $\gamma_i = \alpha_i \sqcup \beta_i$ and $\gamma'_i = \alpha'_i \sqcup \beta'_i$. By the inductive hypothesis $\gamma_i \sqsubseteq \gamma'_i$. Also $\min(s_1, s_2) \geq \min(s'_1, s'_2)$. Hence $\tau_1 \sqcup \tau_2 \sqsubseteq \tau'_1 \sqcup \tau'_2$.

(2) The proof for the second point is analogous to the first point. \square

Corollary C.15. *If $\tau_1 \sqsubseteq \tau$ and $\tau_2 \sqsubseteq \tau$ then $\tau_1 \sqcup \tau_2 \sqsubseteq \tau$.*

Proof. One shows by induction that $\tau \sqsubseteq \tau \sqcap \tau$ and $\tau \sqcup \tau \sqsubseteq \tau$. Then the corollary follows from Lemma C.14 and Lemma C.12. \square

We write $\Gamma \sqsubseteq \Gamma'$ if $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$ and $\Gamma' = x_1 : \tau'_1, \dots, x_n : \tau'_n$ and $\tau_i \sqsubseteq \tau'_i$ for $i = 1, \dots, n$.

Note that if $\alpha \sqcup \beta$ is defined and $\alpha' \sqsubseteq \alpha$ and $\beta' \sqsubseteq \beta$ then $\alpha' \sqcup \beta'$ is also defined. We will often use this observation implicitly.

Lemma C.16. *If $\Gamma' \sqsubseteq \Gamma$ and $\mathcal{T}(\Gamma; t)$ is defined then $\mathcal{T}(\Gamma'; t)$ is defined and $\mathcal{T}(\Gamma'; t) \sqsubseteq \mathcal{T}(\Gamma; t)$.*

Proof. We proceed by induction on the definition of \mathcal{T} .

If $\Gamma' \sqsubseteq \Gamma$ and $\tau' \sqsubseteq \tau$ then $\mathcal{T}(\Gamma, x : \tau; t) = \tau \sqsupseteq \tau' = \mathcal{T}(\Gamma', x : \tau'; t)$.

If $\Gamma' \sqsubseteq \Gamma$ and $\mathcal{T}(\Gamma; ct_1 \dots t_n)$ is defined, then

$$\mathcal{T}(\Gamma; ct_1 \dots t_n) = \mu^{\max(s_1+1, \dots, s_n+1)}$$

where we have $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\mu = d_\mu(\tau_1, \dots, \tau_n)$ and $\text{Def}(c) = d_\mu$ and

$$\mathcal{T}(\Gamma; t_i) = \sigma'_i[d_\mu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m]$$

(we take $s_i = 0$ and $\alpha_j^i = \perp$ if $A \notin \text{TV}(\sigma_i)$, and $\beta_j^i = \perp$ if $B_j \notin \text{TV}(\sigma_i)$) and $\sigma'_i \sqsubseteq \sigma_i$ and $\tau_j = \bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$. By the inductive hypothesis

$$\mathcal{T}(\Gamma'; t_i) \sqsubseteq \mathcal{T}(\Gamma; t_i) = \sigma'_i[d_\mu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m].$$

By Lemma C.3 we have $\mathcal{T}(\Gamma'; t_i) = \rho_i[d_\mu^{s'_i}(\gamma_1^i, \dots, \gamma_m^i)/A][\delta_1^i/B_1, \dots, \delta_m^i/B_m]$ with $\gamma_j^i \sqsubseteq \alpha_j^i$ and $\delta_j^i \sqsubseteq \beta_j^i$ and $s'_i \leq s_i$. Since $\rho_i \sqsubseteq \sigma'_i \sqsubseteq \sigma_i$, by Lemma C.12 we obtain $\rho_i \sqsubseteq \sigma_i$. Let $\tau'_j = \bigsqcup_{i=1}^n (\gamma_j^i \sqcup \delta_j^i)$. Thus $\mathcal{T}(\Gamma'; ct_1 \dots t_n) = \mu_1^{\max(s'_1+1, \dots, s'_n+1)}$ where $\mu_1 = d_\mu(\tau'_1, \dots, \tau'_n)$. By Lemma C.14 we have $\tau'_j \sqsubseteq \tau_j$. Also $\max(s'_1+1, \dots, s'_n+1) \leq \max(s_1+1, \dots, s_n+1)$. Hence $\mathcal{T}(\Gamma'; ct_1 \dots t_n) \sqsubseteq \mathcal{T}(\Gamma; ct_1 \dots t_n)$.

If $\Gamma' \sqsubseteq \Gamma$ and $\mathcal{T}(\Gamma; \lambda x : \alpha. t)$ is defined then $\mathcal{T}(\Gamma; \lambda x : \alpha. t) = \alpha \rightarrow \beta$ and $\mathcal{T}(\Gamma, x : \alpha; t) = \beta$. By the inductive hypothesis $\beta' = \mathcal{T}(\Gamma', x : \alpha; t) \sqsubseteq \beta$. Hence $\mathcal{T}(\Gamma'; \lambda x : \alpha. t) = \alpha \rightarrow \beta' \sqsubseteq \alpha \rightarrow \beta = \mathcal{T}(\Gamma; \lambda x : \alpha. t)$.

If $\Gamma' \sqsubseteq \Gamma$ and $\mathcal{T}(\Gamma; tt') = \beta$ then $\mathcal{T}(\Gamma; t) = \alpha \rightarrow \beta$ and $\mathcal{T}(\Gamma; t') \sqsubseteq \alpha$. By the inductive hypothesis $\mathcal{T}(\Gamma'; t) \sqsubseteq \alpha \rightarrow \beta$ and $\mathcal{T}(\Gamma'; t') \sqsubseteq \mathcal{T}(\Gamma; t')$. Hence $\mathcal{T}(\Gamma'; t) = \alpha' \rightarrow \beta'$ with $\alpha \sqsubseteq \alpha'$ and $\beta' \sqsubseteq \beta$. By Lemma C.12 we have $\mathcal{T}(\Gamma'; t') \sqsubseteq \alpha'$. Hence $\mathcal{T}(\Gamma'; tt') = \beta' \sqsubseteq \beta = \mathcal{T}(\Gamma; tt')$.

If $\Gamma' \sqsubseteq \Gamma$ and $\mathcal{T}(\Gamma; ts) = \tau[s/i]$ then $\mathcal{T}(\Gamma; t) = \forall i. \tau$. By the inductive hypothesis $\mathcal{T}(\Gamma'; t) \sqsubseteq \forall i. \tau$, so $\mathcal{T}(\Gamma'; t) = \forall i. \tau'$ with $\tau' \sqsubseteq \tau$. Hence $\mathcal{T}(\Gamma'; ts) = \tau'[s/i] \sqsubseteq \tau[s/i] = \mathcal{T}(\Gamma; ts)$.

If $\Gamma' \sqsubseteq \Gamma$ and $\mathcal{T}(\Gamma; \Lambda i. t) = \forall i. \tau$ then $\mathcal{T}(\Gamma; t) = \tau$ and $i \notin \text{FSV}(\Gamma)$. By the inductive hypothesis $\mathcal{T}(\Gamma'; t) = \tau' \sqsubseteq \tau$. Without loss of generality $i \notin \text{FSV}(\tau')$. Then $\mathcal{T}(\Gamma'; \Lambda i. t) = \forall i. \tau' \sqsubseteq \mathcal{T}(\Gamma; \Lambda i. t)$.

If $\Gamma' \sqsubseteq \Gamma$ and $\mathcal{T}(\Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\})) = \tau$ and $\mathcal{T}(\Gamma; t) = \nu^s$ then $\nu = d(\vec{\beta})$ and $s \geq 1$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma_k^l[\nu^s/A][\vec{\beta}/\vec{B}]$ and $\mathcal{T}(\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k) = \tau_k$ and $\tau = \bigsqcup_{k=1}^n \tau_k$. By the inductive hypothesis $\mathcal{T}(\Gamma'; t) \sqsubseteq \nu^s$. Hence $\mathcal{T}(\Gamma'; t) = \nu_0^{s'} = d^{s'}(\vec{\beta}')$ with $\beta'_i \sqsubseteq \beta_i$ and $s' \geq s \geq 1$. Let $\gamma_k^l = \sigma_k^l[\nu_0^{s'}/A][\vec{\beta}'/\vec{B}]$. By Lemma C.3 and Lemma C.7 we have $\gamma_k^l \sqsubseteq \delta_k^l$. So by the inductive hypothesis $\mathcal{T}(\Gamma', x_k^1 : \gamma_k^1, \dots, x_k^{n_k} : \gamma_k^{n_k}; t_k) = \tau'_k \sqsubseteq \tau_k$. Let $\tau' = \bigsqcup_{k=1}^n \tau'_k$. Then $\mathcal{T}(\Gamma'; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k\})) = \tau' \sqsubseteq \tau = \mathcal{T}(\Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k\}))$ by Lemma C.14.

Other cases are similar to the ones already considered or follow directly from the inductive hypothesis. \square

Theorem C.17. $\Gamma \vdash t : \tau$ iff $\Gamma \vdash t : \mathcal{T}(\Gamma; t)$ and $\mathcal{T}(\Gamma; t) \sqsubseteq \tau$.

Proof. The implication from right to left follows directly from definitions. For the other direction we proceed by induction on the typing derivation. By Lemma C.11 it suffices to show that $\mathcal{T}(\Gamma; t)$ is defined and $\mathcal{T}(\Gamma; t) \sqsubseteq \tau$.

- If $\Gamma, x : \tau \vdash x : \tau$ then $\mathcal{T}(\Gamma, x : \tau; x) = \tau$.
- If $\Gamma \vdash t : \tau'$ because of $\Gamma \vdash t : \tau$ and $\tau \sqsubseteq \tau'$, then by the inductive hypothesis $\Gamma \vdash t : \mathcal{T}(\Gamma; t)$ and $\mathcal{T}(\Gamma; t) \sqsubseteq \tau$. Hence also $\mathcal{T}(\Gamma; t) \sqsubseteq \tau'$ by Lemma C.12.
- Assume $\Gamma \vdash ct_1 \dots t_n : \rho^{s+1}$ because of $\Gamma \vdash t_k : \sigma_k[\rho^s/A][\vec{\tau}/\vec{B}]$ where $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\text{Def}(c) = d$ and $\rho = d(\vec{\tau})$. Let $\theta_k = \mathcal{T}(\Gamma; t_k)$. By the inductive hypothesis $\Gamma \vdash t_k : \theta_k$ and $\theta_k \sqsubseteq \sigma_k[\rho^s/A][\vec{\tau}/\vec{B}]$. Hence by Lemma C.3 we have $\theta_k = \sigma_k^l[\rho_k^{s_k}/A][\vec{\beta}_k/\vec{B}]$ with $\sigma_k^l \sqsubseteq \sigma_k$ and $\rho_k^{s_k} \sqsubseteq \rho^s$ and $\beta_k^j \sqsubseteq \tau_j$ and $\rho_k = d(\vec{\alpha}_k)$ and $\alpha_k^j \sqsubseteq \tau_j$. We may assume $s_k = 0$ and $\alpha_k^j = \perp$ if $A \notin \text{TV}(\sigma_k)$, and $\beta_k^j = \perp$ if $B_j \notin \text{TV}(\sigma_k)$. Let $\tau_j' = \bigsqcup_{k=1}^m (\alpha_k^j \sqcup \beta_k^j)$. Then $\mathcal{T}(\Gamma; ct_1 \dots t_n) = d^{m(s_1+1, \dots, s_n+1)}(\vec{\tau}')$ where $m = \max$ if d is inductive, and $m = \min$ if d is coinductive. by Lemma C.14 we have $\tau_i' \sqsubseteq \tau_i$. Together with properties of size expressions this implies $\mathcal{T}(\Gamma; ct_1 \dots t_n) \sqsubseteq \rho^{s+1}$.
- Assume $\Gamma \vdash (\lambda x : \alpha. t) : \alpha \rightarrow \beta$ because of $\Gamma, x : \alpha \vdash t : \beta$. By the inductive hypothesis $\beta' = \mathcal{T}(\Gamma, x : \alpha) \sqsubseteq \beta$. Then $\mathcal{T}(\Gamma; \lambda x : \alpha. t) = \alpha \rightarrow \beta' \sqsubseteq \alpha \rightarrow \beta$.
- Assume $\Gamma \vdash tt' : \beta$ because of $\Gamma \vdash t : \alpha \rightarrow \beta$ and $\Gamma \vdash t' : \alpha$. By the inductive hypothesis $\mathcal{T}(\Gamma; t) \sqsubseteq \alpha \rightarrow \beta$ and $\mathcal{T}(\Gamma; t') \sqsubseteq \alpha$. Then $\mathcal{T}(\Gamma; t) = \alpha' \rightarrow \beta'$ with $\alpha \sqsubseteq \alpha'$ and $\beta' \sqsubseteq \beta$. We have $\mathcal{T}(\Gamma; t') \sqsubseteq \alpha'$ by Lemma C.12. Hence $\mathcal{T}(\Gamma; tt') = \beta' \sqsubseteq \beta$.
- Assume $\Gamma \vdash ts : \tau[s/i]$ because of $\Gamma \vdash t : \forall i. \tau$. By the inductive hypothesis $\mathcal{T}(\Gamma; t) \sqsubseteq \forall i. \tau$. Then $\mathcal{T}(\Gamma; t) = \forall i. \tau'$ with $\tau' \sqsubseteq \tau$. Thus $\mathcal{T}(\Gamma; ts) = \tau'[s/i] \sqsubseteq \tau[s/i]$.
- Assume $\Gamma \vdash \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k\}) : \tau$ because of $\Gamma \vdash t : \nu^{s+1}$ and $\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k} \vdash t_k : \tau$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma_k^l[\nu^s/A][\vec{\beta}/\vec{B}]$ and $\nu = d(\vec{\beta})$. By the inductive hypothesis $\mathcal{T}(\Gamma; t) \sqsubseteq \nu^{s+1}$. Hence $\mathcal{T}(\Gamma; t) = d^{s'}(\vec{\beta}')$ with $s' \geq s + 1$ and $\beta'_i \sqsubseteq \beta_i$. Let $\gamma_k^l = \sigma_k^l[d^{s'}(\vec{\tau}')] [\vec{\tau}'/\vec{B}]$. By Lemma C.8 we have $s' \geq s$. So by Lemma C.3 we have $\gamma_k^l \sqsubseteq \delta_k^l$. By the inductive hypothesis $\mathcal{T}(\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k) \sqsubseteq \tau$. By Lemma C.16 we have $\mathcal{T}(\Gamma, x_k^1 : \gamma_k^1, \dots, x_k^{n_k} : \gamma_k^{n_k}; t_k) \sqsubseteq \mathcal{T}(\Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k)$, so $\mathcal{T}(\Gamma, x_k^1 : \gamma_k^1, \dots, x_k^{n_k} : \gamma_k^{n_k}; t_k) = \tau_k \sqsubseteq \tau$. Let $\tau' = \bigsqcup_{k=1}^n \tau_k$. Then $\mathcal{T}(\Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k\})) = \tau'$. By Corollary C.15 we have $\tau' \sqsubseteq \tau$.

- Other cases are analogous to the ones already considered or follow directly from the inductive hypothesis. \square

C.2. Type checking. We now show that type checking in λ^\diamond is coNP-complete. For this purpose we show how to compute the minimal type and how to check subtyping.

The size of a type or a size expression is defined in a natural way as the length of its textual representation. Let U be a partial finite function from the set of size variables to the set of size expression satisfying the *acyclicity condition*: for any choice of j_1, \dots, j_n with $j_1 = i$ and $j_{k+1} \in \text{SV}(U(j_k))$ for $k = 1, \dots, n-1$, we have $j_n \neq i$. In other words, there are no cycles in the directed graph constructed from U by postulating an edge from i to each $j \in \text{SV}(U(i))$. Let S be a set of pairs of size expressions. The size of U (resp. S) is the sum of the sizes of all size expressions in the pairs in U (resp. S). The pair (U, S) is called a *size constraint*. We say that the size constraint (U, S) is *valid* if for every valuation v such that $v(i) = v(U(i))$ holds for all $i \in \text{dom}(U)$, we have $v(s_1) \leq v(s_2)$ for all $(s_1, s_2) \in S$. We sometimes identify the function U with the set of equalities $\{i = U(i) \mid i \in \text{dom}(U)\}$.

The purpose of U is not to express any constraints, but to avoid duplicating size expressions in the inequalities in S . This is in order to avoid exponential blow-up in the size of size constraints.

The size of a finite decorated term t is defined in a natural way, except that for each occurrence of a constant c in t we add the size of $\text{ArgTypes}(c)$ to the size of t .

For a size expression s , by $U(s)$ we denote the size expression s' obtained from s by recursively (i.e. as long as possible) substituting each free occurrence of a size variable $i \in \text{dom}(U)$ with $U(i)$. For example, if $U = \{i_1 = \min(i_2, i_2 + 1), i_2 = s\}$ then $U(\max(i_1, i_1)) = \max(\min(s, s + 1), \min(s, s + 1))$. Because of the acyclicity condition on U the result of this recursive substitution process is well-defined. We extend this in the obvious way to types, terms and contexts. Note that (U, S) is valid iff $U(s_1) \leq U(s_2)$ for all $(s_1, s_2) \in S$.

We now show that it suffices to consider size variable valuations $v : V_S \rightarrow \mathbb{N}$ with the codomain restricted to \mathbb{N} .

Lemma C.18. *If $v(s_1) \leq v(s_2)$ for every $v : V_S \rightarrow \mathbb{N} \cup \{\infty\}$, then $v(s_1) \leq v(s_2)$ for every $v : V_S \rightarrow \Omega$.*

Proof. Assume $v(s_1) > v(s_2)$ for some $v : V_S \rightarrow \Omega$. We show how to construct $v' : V_S \rightarrow \mathbb{N} \cup \{\omega\}$ such that $v'(s_1) > v'(s_2)$. Because $\text{SV}(s_1, s_2)$ is finite, there exist limit ordinals $\iota_1 < \dots < \iota_n < \infty$ such that for each $i \in \text{SV}(s_1, s_2)$ either $v(i) = \infty$ or there are $k, m \in \mathbb{N}$ with $v(i) = \iota_k + m$. Let $M \in \mathbb{N}$ be maximal such that $v(i) = \iota_k + M$ for some i, k . Let N be the maximal nesting of $+1$ in s_1, s_2 , e.g., for a size expression $\max(i + 1, j) + 1$ we have $N = 2$. Let $j_k = k(M + N + 1)$ for $k = 1, \dots, n$. Now it suffices to set $v'(i) = j_k + m$ if $v(i) = \iota_k + m$, and $v'(i) = \infty$ if $v(i) = \infty$. \square

Corollary C.19. *A size constraint (U, S) is valid iff for every $v : V_S \rightarrow \mathbb{N}$ such that $v(i) = v(U(i))$ for $i \in \text{dom}(U)$ we have $v(s_1) \leq v(s_2)$ for all $(s_1, s_2) \in S$.*

Proof. The implication from left to right follows from definitions. The other direction follows from Lemma C.18 and the fact that a non-strict inequality is preserved when taking the limit. \square

Lemma C.20. *The problem of checking whether a size constraint (U, S) is valid is in coNP.*

Proof. The complement of the problem may be reduced to the problem of the satisfiability of a polynomially large formula in quantifier-free Presburger arithmetic, which is in NP [8, 22]. We proceed with the details.

By Corollary C.19 it suffices to consider valuations $v : V_s \rightarrow \mathbb{N}$ with \mathbb{N} as codomain.

Using the identities $\infty + 1 = \max(\infty, s) = \max(s, \infty) = \infty$ and $\min(\infty, s) = \min(s, \infty) = s$ we may simplify each size expression in a linear number of steps to either ∞ or a size expression not containing ∞ . If $U(i) = \infty$ for some $i \in \text{dom}(U)$ then we may substitute ∞ for i in each size expression and set $U(i)$ to undefined. We perform these simplifications for (U, S) as long as possible, obtaining after a polynomial number of steps an equivalent size constraint (U', S') (i.e. such that it is valid iff (U, S) is) such that $U'(i)$ does not contain ∞ and for each $(s_1, s_2) \in S'$ one of the following holds:

- neither s_1 nor s_2 contain ∞ ,
- $s_2 = \infty$ – then (s_1, s_2) may be removed from S because $s_1 \leq \infty$ always holds,
- $s_1 = \infty$ and s_2 does not contain ∞ – then (U', S') is not valid, because then $v(s_2) < \infty$ for $v : V_S \rightarrow \mathbb{N}$.

Hence, we may assume that none of the size expressions in (U, S) contains ∞ .

Thus the answer to our decision problem is negative iff there exists e.g. $(s_1, s_2) \in S$ such that

$$i_1 = s_{i_1} \wedge \dots \wedge i_k = s_{i_k} \wedge s_1 \geq s_2 + 1$$

is satisfiable in natural numbers, where the equalities $i_l = s_{i_l}$ come from U .

Using the identities

$$\begin{aligned} \max(a, b) + 1 &= \max(a + 1, b + 1) \\ \min(a, b) + 1 &= \min(a + 1, b + 1) \end{aligned}$$

we may further normalize the size expressions so that \max and \min never occur within the scope of $+1$.

Hence, it suffices to show that the satisfiability of conjunctions of normalized size expression inequalities is in NP. However, noting that

$$\begin{aligned} \min(a, b) \leq c &\leftrightarrow \exists n. (n \geq a \vee n \geq b) \wedge n \leq c \\ c \leq \min(a, b) &\leftrightarrow \exists n. c \leq n \wedge n \leq a \wedge n \leq b \\ \max(a, b) \leq c &\leftrightarrow \exists n. a \leq n \wedge b \leq n \wedge n \leq c \\ c \leq \max(a, b) &\leftrightarrow \exists n. (n \leq a \vee n \leq b) \wedge n \leq c \end{aligned}$$

this problem may be reduced to satisfiability of a polynomially large formula in quantifier-free Presburger arithmetic. The latter problem is in NP [8, 22]. See also the remark at the end of Section 2.2 in [22]. \square

Lemma C.21. *For any types τ_1, τ_2 there exists $S = \mathcal{S}(\tau_1, \tau_2)$ such that for any U we have: $U(\tau_1) \sqsubseteq U(\tau_2)$ iff (U, S) is valid. Moreover, the size of S is at most polynomial in the size of τ_1, τ_2 .*

Proof. Follows by induction on the definition of \sqsubseteq . \square

Corollary C.22. *Given two types τ_1, τ_2 and a partial finite function U satisfying the acyclicity condition, checking whether $U(\tau_1) \sqsubseteq U(\tau_2)$ is in coNP.*

Proof. Follows from Lemma C.21 and Lemma C.20. \square

Lemma C.23. *Given $k \in \mathbb{N}$, a partial finite function U satisfying the acyclicity condition, and a size expression s , it is decidable in polynomial time whether $U(s) \geq k$.*

Proof. Note that the smallest value of $v(U(s))$ is when $v(i) = 0$ for $i \notin \text{dom}(U)$. So it suffices to evaluate $U(s)$ with all size variables set to 0 and check whether the result is at least k . This may be done in polynomial time. \square

Lemma C.24. *Given a finite context Γ and a term t , one may compute in polynomial time a triple (U, S, τ) of polynomial size satisfying:*

- (U, S) is valid iff $\mathcal{T}(\Gamma; t)$ is defined,
- if $\mathcal{T}(\Gamma; t)$ is defined then $U(\tau) = \mathcal{T}(\Gamma; t)$.

Proof. We semi-informally describe an algorithm to compute (U, S, τ) by the following definition of a recursive function $\mathcal{T}'(U_0; \Gamma; t)$. To obtain the desired triple one takes $U_0 = \emptyset$.

- $\mathcal{T}'(U_0; \Gamma, x : \tau; x) = (U_0, \emptyset, \tau)$.
- $\mathcal{T}'(U_0; \Gamma; ct_1 \dots t_n) = (U, S, \theta)$ if $\theta = \mu^{\max(s_1, \dots, s_n)+1}$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\mu = d_\mu(\tau_1, \dots, \tau_m)$ and $\text{Def}(c) = d_\mu$ and $\mathcal{T}'(U_0; \Gamma; t_i) = (U_i, S_i, \theta_i)$ and

$$\theta_i = \sigma'_i[d_\mu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m]$$

(we take $s_i = 0$ and $\alpha_j^i = \perp$ if $A \notin \text{TV}(\sigma_i)$, and $\beta_j^i = \perp$ if $B_j \notin \text{TV}(\sigma_i)$; if some θ_i does not have the desired form then the present case does not apply) and $\tau_j = \bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$ (if $\bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$ is not defined then the present case does not apply) and $U = \bigcup_{i=0}^n U_i$ and $S = \bigcup_{i=1}^n S_i \cup \mathcal{S}(\sigma'_i, \sigma_i)$.

- $\mathcal{T}'(U_0; \Gamma; ct_1 \dots t_n) = (U, S, \theta)$ if $\theta = \nu^{\min(s_1, \dots, s_n)+1}$ and $\text{ArgTypes}(c) = (\sigma_1, \dots, \sigma_n)$ and $\nu = d_\nu(\tau_1, \dots, \tau_m)$ and $\text{Def}(c) = d_\nu$ and $\mathcal{T}'(U_0; \Gamma; t_i) = (U_i, S_i, \theta_i)$ and

$$\theta_i = \sigma'_i[d_\nu^{s_i}(\alpha_1^i, \dots, \alpha_m^i)/A][\beta_1^i/B_1, \dots, \beta_m^i/B_m]$$

(we take $s_i = 0$ and $\alpha_j^i = \perp$ if $A \notin \text{TV}(\sigma_i)$, and $\beta_j^i = \perp$ if $B_j \notin \text{TV}(\sigma_i)$; if some θ_i does not have the desired form then the present case does not apply) and $\tau_j = \bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$ (if $\bigsqcup_{i=1}^n (\alpha_j^i \sqcup \beta_j^i)$ is not defined then the present case does not apply) and $U = \bigcup_{i=0}^n U_i$ and $S = \bigcup_{i=1}^n S_i \cup \mathcal{S}(\sigma'_i, \sigma_i)$.

- $\mathcal{T}'(U_0; \Gamma; \lambda x : \alpha.t) = (U, S, \alpha \rightarrow \beta)$ if $\mathcal{T}'(U_0; \Gamma, x : \alpha; t) = (U, S, \beta)$.
- $\mathcal{T}'(U_0; \Gamma; tt')$ if $\mathcal{T}(U_0; \Gamma; t) = (U_1, S_1, \alpha \rightarrow \beta)$ and $\mathcal{T}'(U_0; \Gamma; t') = (U_2, S_2, \alpha')$ and $U = U_1 \cup U_2$ and $S = S_1 \cup S_2 \cup \mathcal{S}(\alpha', \alpha)$.
- $\mathcal{T}'(U_0; \Gamma; ts) = (U \cup \{i = s\}, S, \tau)$ if $\mathcal{T}'(U_0; \Gamma; t) = (U, S, \forall i. \tau)$ with i fresh.
- $\mathcal{T}'(U_0; \Gamma; \Lambda i. t) = (U, S, \forall i. \tau)$ if $\mathcal{T}'(U_0; \Gamma; t) = (U, S, \tau)$ and $i \notin \text{FSV}(\Gamma)$.
- $\mathcal{T}'(U'; \Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\})) = (U, S, \tau)$ if $\mathcal{T}'(U'; \Gamma; t) = (U_0, S_0, \mu^s)$ and $\mu = d(\vec{\beta})$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma_k^l[\mu^i/A][\vec{\beta}/\vec{B}]$ with i fresh and $\mathcal{T}'(U' \cup \{i = \underline{s}\}; \Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k) = (U_k, S_k, \tau_k)$ and $\tau = \bigsqcup_{k=1}^n \tau_k$ and $U = \bigcup_{i=0}^n U_i$ and $S = \bigcup_{i=0}^n S_i$.
- $\mathcal{T}'(U'; \Gamma; \text{case}(t; \{c_k \vec{x}_k \Rightarrow t_k \mid k = 1, \dots, n\})) = (U, S, \tau)$ if $\mathcal{T}'(U'; \Gamma; t) = (U_0, S_0, \nu^s)$ and $\nu = d(\vec{\beta})$ and $s \geq 1$ and $\text{ArgTypes}(c_k) = (\sigma_k^1, \dots, \sigma_k^{n_k})$ and $\delta_k^l = \sigma_k^l[\nu^i/A][\vec{\beta}/\vec{B}]$ with i fresh and $\mathcal{T}'(U' \cup \{i = \bar{s}\}; \Gamma, x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}; t_k) = (U_k, S_k, \tau_k)$ and $\tau = \bigsqcup_{k=1}^n \tau_k$ and $U = \bigcup_{i=0}^n U_i$ and $S = \bigcup_{i=0}^n S_i$.
- $\mathcal{T}'(U_0; \Gamma; \text{fix } f : (\forall j_1 \dots j_m. \mu \rightarrow \tau). t) = (U, S, \forall j_1 \dots j_m. \mu \rightarrow \tau)$ if

$$\mathcal{T}'(U_0; \Gamma, f : \forall j_1 \dots j_m. \mu^i \rightarrow \tau; t) = (U, S_0, \theta)$$

and $i \notin \text{FSV}(\Gamma, \mu, \tau, j_1, \dots, j_m)$ and $S = S_0 \cup \mathcal{S}(\theta, \forall j_1 \dots j_m. \mu \rightarrow \tau)$.

- $\mathcal{T}'(U_0; \Gamma; \text{cofix } f : \tau.t) = (U, S, \tau)$ if

$$\mathcal{T}'(U_0; \Gamma, f : \text{chgtgt}(\tau, \nu^{\min(s, j)}); t) = (U, S_0, \theta)$$

- and $\text{tgt}(\tau) = \nu^s$ and $j \notin \text{FSV}(\Gamma)$ and $j \notin \text{SV}(\tau)$ and $S = S_0 \cup \mathcal{S}(\theta, \text{chgtgt}(\tau, \nu^{\min(s, j+1)}))$.
- Otherwise, if none of the above cases hold, we define $\mathcal{T}'(U_0; \Gamma; t) = (U_0, \{1 \leq 0\}, \perp)$ with \perp and arbitrary fixed type.

First note that if $U' \supseteq U$ where the new size variables from $\text{dom}(U') \setminus \text{dom}(U)$ do not occur in S or τ then: (1) (U, S) is valid iff (U', S) is valid, and (2) $U(\tau) = U'(\tau)$. Note also that when forming a sum $U = \bigcup_{i=1}^n U_i$ in the above definition, the function (set of equations) U is well-defined and satisfies the acyclicity condition because the left-hand side variable i in each newly added equation $i = s$ is always chosen to be fresh. Using these observations one shows by induction that if $\mathcal{T}'(U'; \Gamma; t) = (U, S, \tau)$ then:

- (U, S) is valid iff $\mathcal{T}(U'(\Gamma); U'(t))$ is defined,
- if $\mathcal{T}(U'(\Gamma); U'(t))$ is defined then we have $U(\tau) = \mathcal{T}(U'(\Gamma); U'(t))$.

It remains to check that the algorithm implicit in the definition of \mathcal{T}' is polynomial. Let N be the initial size of the input (i.e. the size of Γ, t). The total number of calls to \mathcal{T}' is proportional to N , because in each immediate recursive call in the definition of $\mathcal{T}'(U_0; \Gamma; t)$ different disjoint proper subterms of t are given as the third argument.

In each immediate recursive call the size of the context Γ grows by at most $O(N^2)$. Indeed, there are essentially two possibilities of what we add to the context Γ .

- (1) We add $x : \alpha$ for the case of lambda abstraction $\lambda x : \alpha.t'$. Then α occurs in the original term t , so the size of Γ grows by at most N .
- (2) We add e.g. $x_k^1 : \delta_k^1, \dots, x_k^{n_k} : \delta_k^{n_k}$ where $\delta_k^j = \sigma_k^j[\mu^i/A][\vec{\beta}/B]$ or the case of a case-term. Then μ and $\vec{\beta}$ occur in the original term t , and σ_k^j is an argument type for the constructor c_k which occurs in t (so the size of σ_k^j counts towards the size of t). Hence the total size of $\sigma_k^1, \dots, \sigma_k^{n_k}$ is $\leq N$, and thus so is the total number of occurrences of A, \vec{B} in $\sigma_k^1, \dots, \sigma_k^{n_k}$. The size of each of $\mu, \vec{\beta}$ is $\leq N$. Therefore, the total size of $\delta_k^1, \dots, \delta_k^{n_k}$ is at most $N^2 + N$.

Hence, the size of the context at any given call to \mathcal{T}' (during the whole run of the algorithm) is at most $O(N^3)$. Let (U, S, τ) denote the result of calling \mathcal{T}' . At the leaves of the computation tree (i.e. when there are no more immediate recursive calls) the type τ is taken from the context, so its size is at most $O(N^3)$. At internal nodes, the size of τ is equal to at most the sum of sizes of the types returned by immediate recursive calls (note that the size of $\alpha \sqcup \beta$ is equal to at most the sum of sizes of α and β plus a constant), plus possibly the size of a type occurring in t (which is $\leq N$), plus possibly $O(N)$. Hence, each call to \mathcal{T}' contributes at most $O(N^3)$ towards the size of the final result type. Since there are $O(N)$ calls in total, for any given call the result type τ of this call has size at most $O(N^4)$. Now we count the final size of S . At the leaves of the computation tree $S = \emptyset$, and at each internal node we add at most $O(N)$ sets $\mathcal{S}(\alpha, \beta)$ where each of α, β is either a subtype of a type returned by an immediate recursive call or of the term t . So the size of α, β is polynomial in N , and thus so is the size of $\mathcal{S}(\alpha, \beta)$ by Lemma C.21. Hence, the total final size of S is polynomial in N . To count the total final size of U , note that we may consider it to be a mutable global variable which at each call is modified by adding at most one equation of polynomial size (because the right-hand side size expression has size proportional to the size of a size expression occurring in a type returned by one of the immediate recursive calls). Thus the total final size of U is polynomial in N .

We have thus shown that the computed triple (U, S, τ) has polynomial size. Note that in each of the calls to \mathcal{T}' , the computation time (not counting the immediate recursive calls) is

proportional to the size of the returned triple, and is thus polynomial (we need Lemma C.23 to decide in polynomial time if $U(s_0) \geq 1$ in the third-last point in the definition of \mathcal{T}'). Hence, the whole running time is polynomial. \square

Theorem 6.5. *Type checking in the system λ^\diamond is coNP-complete. More precisely, given Γ, t, τ the problem of checking whether $\Gamma \vdash t : \tau$ is coNP-complete.*

Proof. It follows from Theorem C.17, Lemma C.24, Lemma C.20 and Corollary C.22 that the problem is in coNP.

To show that the problem is coNP-hard we reduce the problem of unsatisfiability of 3-CNF boolean formulas, which is coNP-hard. We show how to construct in polynomial time an inequality $s_1 \leq s_2$ of size expressions which is equisatisfiable with a given 3-CNF boolean formula φ . For concreteness assume φ is

$$(x \vee \neg y \vee z) \wedge (x \vee \neg z \vee y).$$

This formula is translated to the inequality $s_1 \leq s_2$ where

$$\begin{aligned} s_1 &= \max(\min(x, \bar{y}, z) + 1, \min(x, \bar{z}, y) + 1, 1, \\ &\quad \min(x, \bar{x}) + 1, \min(y, \bar{y}) + 1, \min(z, \bar{z}) + 1) \\ s_2 &= \min(1, \max(x, \bar{x}), \max(y, \bar{y}), \max(z, \bar{z})) \end{aligned}$$

and $\bar{x}, \bar{y}, \bar{z}$ are fresh variables intended to represent the negations of x, y, z respectively.

Let v with $\text{codom}(v) = \{\top, \perp\}$ be a satisfying valuation for φ . Define \bar{v} with $\text{codom}(\bar{v}) = \{0, 1\}$ by $\bar{v}(i) = 0$ if $v(i) = \top$, and $\bar{v}(i) = 1$ if $v(i) = \perp$, and $\bar{v}(\bar{i}) = 1 - v(i)$, for any variable i . Then $\bar{v}(\max(i, \bar{i})) = \bar{v}(\min(i, \bar{i}) + 1) = 1$ for any variable i , and $\bar{v}(\min(x, \bar{y}, z)) = \bar{v}(\min(x, \bar{z}, y)) = 0$. Hence $\bar{v}(s_1) \leq \bar{v}(s_2)$.

Let \bar{v} be a satisfying valuation for $s_1 \leq s_2$. The inequality $s_1 \leq s_2$ is equivalent to the following conjunction of inequalities:

$$\begin{aligned} \min(x, \bar{y}, z) + 1 &\leq 1 \wedge \min(x, \bar{y}, z) + 1 \leq \max(x, \bar{x}) \wedge \\ \min(x, \bar{y}, z) + 1 &\leq \max(y, \bar{y}) \wedge \min(x, \bar{y}, z) + 1 \leq \max(z, \bar{z}) \wedge \\ \min(x, \bar{z}, y) + 1 &\leq 1 \wedge \min(x, \bar{z}, y) + 1 \leq \max(x, \bar{x}) \wedge \\ \min(x, \bar{z}, y) + 1 &\leq \max(y, \bar{y}) \wedge \min(x, \bar{z}, y) + 1 \leq \max(z, \bar{z}) \wedge \\ 1 &\leq 1 \wedge 1 \leq \max(x, \bar{x}) \wedge 1 \leq \max(y, \bar{y}) \wedge 1 \leq \max(z, \bar{z}) \wedge \\ \min(x, \bar{x}) + 1 &\leq 1 \wedge \min(x, \bar{x}) + 1 \leq \max(x, \bar{x}) \wedge \min(x, \bar{x}) + 1 \leq \max(y, \bar{y}) \wedge \\ \min(x, \bar{x}) + 1 &\leq \max(z, \bar{z}) \wedge \\ \min(y, \bar{y}) + 1 &\leq 1 \wedge \min(y, \bar{y}) + 1 \leq \max(x, \bar{x}) \wedge \min(y, \bar{y}) + 1 \leq \max(y, \bar{y}) \wedge \\ \min(y, \bar{y}) + 1 &\leq \max(z, \bar{z}) \wedge \\ \min(z, \bar{z}) + 1 &\leq 1 \wedge \min(z, \bar{z}) + 1 \leq \max(x, \bar{x}) \wedge \min(z, \bar{z}) + 1 \leq \max(y, \bar{y}) \wedge \\ \min(z, \bar{z}) + 1 &\leq \max(z, \bar{z}). \end{aligned}$$

For each variable i , since $1 \leq \max(i, \bar{i})$ and $\min(i, \bar{i}) + 1 \leq 1$ occur in this conjunction, we conclude that exactly one of $\bar{v}(i), \bar{v}(\bar{i})$ is zero and the other one is nonzero. Define v with $\text{codom}(v) = \{\top, \perp\}$ by $v(i) = \top$ if $\bar{v}(i) = 0$, and $v(i) = \perp$ if $\bar{v}(i) \neq 0$. We have $\bar{v}(\min(x, \bar{y}, z)) = \bar{v}(\min(x, \bar{z}, y)) = 0$ because of the inequalities $\min(x, \bar{y}, z) + 1 \leq 1$ and $\min(x, \bar{z}, y) + 1 \leq 1$. This implies that v is a satisfying valuation for φ .

Hence for every 3-CNF boolean formula φ there exists an equisatisfiable inequality $s_1 \leq s_2$ of size expressions which may be computed in polynomial time. So φ is unsatisfiable iff $s_1 > s_2$ is valid. Because $v(s_2) \neq \infty$ for any valuation v , the inequality $s_1 > s_2$ is equivalent to $s_1 \geq s_2 + 1$. Therefore φ is unsatisfiable iff $x : \nu^{s_1}, f : \nu^{s_2+1} \rightarrow \mu^0 \vdash fx : \mu^0$ for some fixed ν, μ . \square

Remark C.25. The use of the set of equations U is necessary to avoid an exponential blow-up in the size of size constraints. For instance, consider $\Gamma = f : \forall i. \mu^i \rightarrow \mu^i$ and define $t_0 = f$, $t_{n+1} = \Lambda i. t_n \max(i, i)$. Let $s_0 = i$ and $s_{n+1} = \max(s_n, s_n)$. We have $\mathcal{T}(\Gamma; t_n) = \forall i. \mu^{s_n} \rightarrow \mu^{s_n}$. The size of s_n is proportional to 2^n while the size of t_n is proportional to n .

Similarly, suppose μ has two constructors $c_1 : \mu \rightarrow \mu$ and $c_2 : \mu \rightarrow \mu$. Let $t_0 = x$ and $t_{n+1} = \text{case}(t_n; \{c_1 y \Rightarrow y, c_2 y \Rightarrow y\})$. Let $s_n = 0 + 1 + 1 + \dots + 1$ where 1 occurs n times. By induction on n one shows $\mathcal{T}(x : \mu^{s_n}; t_n) = \mu^{s'_n}$ where $s'_0 = 0$ and $s'_{n+1} = \max(s'_n, s'_n)$. The size of s'_n is proportional to 2^n , while the sizes of t_n, s_n are proportional to n .