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 \( \nu \) are possibly infinite terms. Each finite term of type \( \nu \) containing fixpoint operators then “unfolds” to a possibly infinite term without fixpoint operators in the interpretation of \( \nu \). For instance, one would interpret the type of binary streams as the set of infinite terms of the form \( b_1 :: b_2 :: \ldots \) 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.
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
\[ \text{Tree}_2 = \text{CoInd}\{ c_1 : \text{List}(\text{Tree}_2) \to \text{Tree}_2, c_2 : (\text{Nat} \to \text{Tree}_2) \to \text{Tree}_2 \} \]
which intuitively specifies that each element of Tree$_2$ is a possibly infinite term which has one of the forms:

- \( c_1(t_1 :: t_2 :: \ldots :: t_n :: \text{nil}) \) 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 \( \tau \) as a subset \( [\tau] \) of the set \( T^\infty \) 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 \( [\text{Strm}] \) of the coinductive type Strm of streams of natural numbers with a single constructor cons : Nat \( \to \) Strm \( \to \) Strm consists of all infinite terms of the form cons \( n_0(\text{cons} \ n_1(\ldots)) \) where \( n_k \in [\text{Nat}] \) for \( k \in \mathbb{N} \). The interpretation \( [\text{Strm} \to \text{Strm}] \) of an arrow type Strm \( \to \) Strm is the set of all terms \( t \) such that for every \( u \in [\text{Strm}] \) there is \( u' \in [\text{Strm}] \) with \( tu \to^\infty u' \), where \( \to^\infty \) 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 \( t \to^\infty t_n \in [\nu]^n \) for each \( n \in \mathbb{N} \) then there is \( t' \) with \( t \to^\infty t' \in [\nu] \), where \( \nu \) is a coinductive type and \( [\nu]^n \) is the set of approximations of size \( n \) of the (typically infinite) objects of type \( \nu \) (i.e. of the terms in \([\nu] \)).

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 $\tau$ in our type system, then there is $t' \in [\tau]$ 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 Isihara’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 $\lambda$-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 \textit{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 familiarly with the lambda calculus [5] and basic notions such as \(\alpha\)-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 \(V\) of variables, and a countable set \(C\) of constructors. The set \(T^\infty\) of all finite and infinite terms \(t\) is given by

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

where \(x \in V\) and \(c, c_k \in C\). 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 \(T^\infty\) 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 \(\alpha\)-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:

\[
(\lambda x.t)t' \rightarrow_{\beta} t'[x / x] \\
\text{case}(c_k\vec{u}; \{c_k\vec{x} \Rightarrow t_k\}) \rightarrow_{\iota} t_k[\vec{u} / \vec{x}]
\]

In the \(\iota\)-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, case\((c_1t_2; \{cxy \Rightarrow x, dxy \Rightarrow y\}) \rightarrow_{\iota} t_1\) (assuming \(c \neq d\)), but case\((c'1t_2; \{cxy \Rightarrow x, dxy \Rightarrow y\})\) and case\((c_1t_2; \{cxy \Rightarrow x, cxy \Rightarrow y\})\) do not have \(\iota\)-reducts (assuming \(c' \notin \{c, d\}\)). We usually write \(t \rightarrow^* t'\) to denote a finitary reduction \(t \rightarrow_{\beta} t'\).

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

\[
\frac{
t \rightarrow^* x}{t \rightarrow^\infty x} \quad \frac{
t \rightarrow^* c}{t \rightarrow^\infty c}
\]

\[
\frac{
t \rightarrow^* \lambda x.r \quad r \rightarrow^\infty r'}{t \rightarrow^\infty \lambda x.r'} \quad \frac{
t \rightarrow^* r_1r_2 \quad r_k \rightarrow^\infty r_k'}{t \rightarrow^\infty r_1' r_2'}
\]

\[
\frac{
t \rightarrow^* \text{case}(r; \{c_k\vec{x} \Rightarrow r_k\}) \quad r \rightarrow^\infty r' \quad r_k \rightarrow^\infty r_k'}{t \rightarrow^\infty \text{case}(r'; \{c_k\vec{x} \Rightarrow r_k'\})}
\]

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 \(\rightarrow^\infty\) is that the depth at which a redex is contracted should tend to infinity.
This is achieved by defining $\rightarrow^\infty$ 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 $\rightarrow^\infty$ 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 $t[t_2/x] \rightarrow^* t'[t_2'/x]$. □

**Lemma 2.3.** If $t \rightarrow^\infty t' \rightarrow^\beta_\ell t''$ then $t \rightarrow^\infty t''$.

*Proof.* Induction on $t' \rightarrow^\beta_\ell t''$, using Lemma 2.2. □

**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. □

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 $\rightarrow^\infty$, but replacing $\rightarrow^*$ with $\rightarrow^\infty$ and $\rightarrow^\infty$ with $\rightarrow^{2\infty}$ in Definition 2.1.

We may consider $\rightarrow^\infty$ (resp. $\rightarrow^{2\infty}$) as defining a strongly convergent ordinal-indexed reduction sequence [27] of length at most $\omega$ (resp. $\omega^2$), 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 $t \rightarrow^\infty t'$.


The system of $\beta_\ell$-reductions on infinitary terms $T^\infty$ 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 $c_1, \ldots, c_n$ and each sequence of natural numbers $k_1, \ldots, k_n$, a symbol $\text{case}_{c_1, \ldots, c_n}^{k_1, \ldots, k_n}$ 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 $\text{case}_{c_1, \ldots, c_n}^{k_1, \ldots, k_n}$ and each $i = 1, \ldots, n$:

$$\text{case}_{c_1, \ldots, c_n}^{k_1, \ldots, k_n}(\text{app}(\ldots(\text{app}(c_i, X_1), X_2)\ldots), X_{k_i}),$$

$$\quad [x_1, \ldots, x_{k_i}]Z_1(x_1, \ldots, x_{k_1}), \ldots, [x_1, \ldots, x_{k_n}]Z_n(x_1, \ldots, x_{k_n})$$

$$\rightarrow Z_i(X_1, \ldots, X_{k_i})$$

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

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 $\rightarrow^{\infty}$ 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. □

**Definition 2.9.** A term $t$ is root-active if for every $t'$ with $t \rightarrow^{\infty} t'$ there is a $\beta\iota$-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 $t \sim_{\mathcal{U}} t'$ then $t \sim_{\mathcal{U}} t''$.

**Proof.** Follows from [32, Proposition 4.12]. □

**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]. □

**Theorem 2.12.** The relation of infinitary reduction $\rightarrow^{\infty}$ 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]. □
### 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., \((i + 3) + 1\) for \((i + 1) + 1\), 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 \(d, d', d_1, \ldots\). Types \(\tau, \alpha, \beta\) are defined by:

\[ \tau ::= A \mid d(s(\tau_1, \ldots, \tau_n)) \mid \tau_1 \to \tau_2 \mid \forall i. \tau \]

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 \(\tau\) is **strictly positive** if one of the following holds:

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

By \(SV(s)\) (resp. \(SV(\tau)\)) we denote the set of all size variables occurring in \(s\) (resp. \(\tau\)). By \(TV(\tau)\) we denote the set of all type variables occurring in \(\tau\). By \(FSV(\tau)\) we denote the set of all free size variables occurring in \(\tau\) (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 \(\tau[\alpha_1/A_1, \ldots, \alpha_n/A_n]\) 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

\[ d(B_1, \ldots, B_n) = \text{CoInd}(A)\{c_k : \sigma^k \mid k = 1, \ldots, m\} \]

where \(A\) is the recursive type variable, and \(B_1, \ldots, B_n\) are the parameter type variables, and \(m > 0\), and \(c_k\) is the \(k\)th constructor, and \(\sigma^k\) is the \(k\)th constructor's \(l\)th argument type, and the following is satisfied:

- \(\sigma^k_l\) are all strictly positive,
- \(TV(\sigma^k) \subseteq \{A, B_1, \ldots, B_n\}\),
- \(FSV(\sigma^k) = \emptyset\).

An **inductive definition** is specified analogously, but using \(\text{Ind}\) instead of \(\text{CoInd}\).

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 $(\sigma^1_k, \ldots, \sigma^n_k)$: 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

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

Formally, here $\sigma^1_1 = A$, $\sigma^1_2 = B$, and $\sigma^2_2 = A$.

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

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

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

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

**Definition 3.5.** An expression of the form $d(\tau_1, \ldots, \tau_n)$ is a (co)inductive type, depending on whether $d$ is an inductive or coinductive definition. A type of the form $d^s(\tau_1, \ldots, \tau_n)$ 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 $\rho$.

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

By default, $d_\nu$ denotes a coinductive and $d_\mu$ an inductive definition. We use $\mu$ for inductive and $\nu$ for coinductive types, and $\rho$ for (co)inductive types when it is not important if it is inductive or coinductive. Analogously, we use $\mu^s$, $\nu^s$, $\rho^s$ for decorated (co)inductive types (with size $s$). We often omit the superscript $\infty$ in $\rho^\infty$, overloading the notation.

Intuitively, $\mu^s$ denotes the type of objects of an inductive type $\mu$ which have size at most $s$, and $\nu^s$ denotes the type of objects of a coinductive type $\nu$ which have size at least $s$, i.e., considered up to depth $s$ they represent a valid object of type $\nu$. For a stream $\nu = \text{Strm}$, the type $\text{Strm}^s$ 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. $\forall i. \text{Strm}^i \to \text{Strm}^s$ 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 $\tau$. 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:

\[ \text{BTree} = \text{CoInd}\{\text{bnode} : \text{Nat} \to \text{BTree} \to \text{BTree} \to \text{BTree}\} \]

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

\[ \text{FTree} = \text{CoInd}\{\text{fnode} : \text{Nat} \to \text{List(FTree)} \to \text{FTree}\} \]

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:

\[ \text{Tree} = \text{CoInd}\{\text{node} : \text{Nat} \to (\text{Nat} \to \text{Tree}) \to \text{Tree}\} \]

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 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:

\[ \text{SPi}(B) = \text{Ind}\{\text{get} : (\text{Nat} \to \text{SPi}(B)) \to \text{SPi}(B), \text{put} : \text{Nat} \to B \to \text{SPi}(B)\} \]

\[ \text{SP} = \text{CoInd}\{\text{out} : \text{SPi}(\text{SP}) \to \text{SP}\} \]

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:

\[ \text{Odd} = \text{Ind}\{S_o : \text{Even} \to \text{Odd}\} \]

\[ \text{Even} = \text{Ind}\{0 : \text{Even}, S_e : \text{Odd} \to \text{Even}\} \]

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

\[ \text{Odd}_0(B) = \text{Ind}\{S_o : B \to \text{Odd}_0(B)\} \]

\[ \text{Even} = \text{Ind}\{0 : \text{Even}, S_e : \text{Odd}_0(\text{Even}) \to \text{Even}\} \]

Now the type Odd is represented by Odd_0(\text{Even}).

In the rest of this paper by “induction on a type \( \tau \)” we mean induction on the lexicographic product of:
• the multiset extension of the well-founded order < on (co)inductive definitions occurring in the type, and
• the size of the type.
In this order, if \( c \in \text{Constr}(\rho) \) with \( \text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_m) \) then each \( \sigma_k \) is smaller than \( \rho \).

4. Rewriting semantics

In this section we define our rewriting semantics. More precisely, we define an interpretation \([\tau] \subseteq T^\infty\) for each type \( \tau \).

By \( \infty \) we denote a sufficiently large ordinal (see Definition 4.1), and by \( \Omega \) we denote the set of all ordinals not greater than \( \infty \). A size variable valuation is a function \( v : V_S \rightarrow \Omega \).

Any size variable valuation \( v \) extends in a natural way to a function from size expressions to \( \Omega \). 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 \( \infty \) with \( \infty \), e.g., \( \infty + 1 \) denotes the ordinal \( \infty \).

**Definition 4.1.** We interpret types as subsets of \( T^\infty \) and \( \Omega \) is the least fixedpoint in \( P \).

In this section we define our rewriting semantics. More precisely, we define an interpretation \( \rho \) than \( c \) in the type, and

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

Any size variable valuation \( v \) extends in a natural way to a function from size expressions to \( \Omega \). 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 \( \infty \) with \( \infty \), e.g., \( \infty + 1 \) denotes the ordinal \( \infty \).

We simultaneously also define valuation approximations \([\rho]_{\xi,v}^\kappa\) and \([d]_{\xi,v}^\kappa\).

- Let \( d(B_1, \ldots, B_n) = (\text{CoInd}(A)\{c_k : \sigma_k \rightarrow d(\bar{B})\}) \) be a (co)inductive definition. We define a function \( \Phi_{d,\xi,v} : P(T^\infty) \rightarrow P(T^\infty) \) so that \( \Phi_{d,\xi,v}(X) \) for \( X \subseteq T^\infty \) contains all terms of the form \( c_k \bar{t}_k \ldots \bar{t}_{nk} \) such that \( \bar{t}_k \in [\sigma_k]_{\xi,v}^{\kappa} \) for \( l = 1, \ldots, n_k \).

For a coinductive definition \( d \) and an ordinal \( \kappa \in \Omega \) we define the valuation approximation \([d]_{\xi,v}^\kappa \subseteq T^\infty\) as follows:

- \([d^0]_{\xi,v}^\kappa = T^\infty\),
- \([d^\kappa+1]_{\xi,v}^\kappa = \Phi_{d,\xi,v}([d^\kappa]_{\xi,v}^\kappa)\),
- \([d^\kappa]_{\xi,v}^\kappa = \bigcap_{\kappa < \kappa} [d^\kappa]_{\xi,v}^\kappa\) if \( \kappa \) is a limit ordinal.

For an inductive definition \( d \) and an ordinal \( \kappa \in \Omega \) we define the valuation approximation \([d]_{\xi,v}^\kappa \subseteq T^\infty\) by:

- \([d^0]_{\xi,v}^\kappa = \emptyset\),
- \([d^\kappa+1]_{\xi,v}^\kappa = \Phi_{d,\xi,v}([d^\kappa]_{\xi,v}^\kappa)\),
- \([d^\kappa]_{\xi,v}^\kappa = \bigcup_{\kappa < \kappa} [d^\kappa]_{\xi,v}^\kappa\) if \( \kappa \) is a limit ordinal.

- \([\beta]_{\xi,v}^\kappa = [d]_{\xi,v}^\kappa \) where \( \rho = d(\bar{a}) \) is a (co)inductive type, \( Y_j = [\alpha_j]_{\xi,v}^\kappa \), and \( \bar{B} \) are the parameter type variables of \( d \).

- \([\rho^\sigma]_{\xi,v}^\kappa = [\rho]_{\xi,v}^{\sigma(s)}\).
- \([A]_{\xi,v}^\kappa = [\xi](A)\).
- \( t \in [\bar{V}_i]_{\xi,v}^\kappa \) if \( i \notin \text{FSV}(t) \) and for every \( \kappa \in \Omega \) there is \( t' \) with \( t \rightarrow^{\kappa} t' \in [\tau]_{\xi,v}[\kappa/i] \).
- \( t \in [\alpha \rightarrow \beta]_{\xi,v}^\kappa \) if for every \( r \in [\alpha]_{\xi,v}^\kappa \) there is \( t' \) with \( tr \rightarrow^{\kappa} t' \in [\beta]_{\xi,v}^\kappa \).
For a closed type $\tau$ the valuation $[\tau]_{\xi,v}$ does not depend on $\xi$, so we simply write $[\tau]_v$ instead. Whenever we omit the type variable valuation we implicitly assume the type to be closed.

In general, the interpretation $[\tau]$ of a type $\tau$ 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 $\tau$ 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 $[\tau]$ contains only normal forms.

Thus, we do not show infinitary weak normalization for terms having function types. Nonetheless, our interpretation of $t \in [\tau_1 \rightarrow \tau_2]$ is very natural and ensures productivity of $t$ regarded as a function: we require that for $u \in [\tau_1]$ there is $u' \in [\tau_2]$ with $tu \rightarrow^\infty u'$. Intuitively, this means that for any $u \in [\tau_1]$ the application $tu$ reduces “in the limit” to a term $u' \in [\tau_2]$, 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:

\[
\begin{align*}
\text{Nat} & \quad= \quad \text{Ind}\{0 : \text{Nat}, S : \text{Nat} \rightarrow \text{Nat}\} \\
\text{Strm} & \quad= \quad \text{CoInd}\{\text{cons} : \text{Nat} \rightarrow \text{Strm} \rightarrow \text{Strm}\}
\end{align*}
\]

The elements of $[\text{Nat}]$ are the terms: $0, S(0), S(S(0)), \ldots$. 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 $[\text{Strm}]$ are all infinite terms of the form $n_1 :: n_2 :: n_3 :: \ldots$ where $n_i \in [\text{Nat}]$.

Consider the term

\[t_1 = \lambda t.\text{case}(t; \{\text{cons} \, x \, y \Rightarrow y\})\]

We have $t_1 \in [\text{Strm} \rightarrow \text{Strm}]$. Indeed, let $t \in [\text{Strm}]$. Then $t = n :: t'$ with $n \in [\text{Nat}]$ and $t' \in [\text{Strm}]$. Thus $t_1(t) \rightarrow \text{case}(n :: t'; \{\text{cons} \, x \, y \Rightarrow y\}) \rightarrow t' \in [\text{Strm}]$.

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

\[
\begin{align*}
\text{BTree} & \quad= \quad \text{CoInd}\{\text{bnode} : \text{Nat} \rightarrow \text{BTree} \rightarrow \text{BTree} \rightarrow \text{BTree}\} \\
\text{FTree} & \quad= \quad \text{CoInd}\{\text{fnode} : \text{Nat} \rightarrow \text{List}(\text{FTree}) \rightarrow \text{FTree}\} \\
\text{Tree} & \quad= \quad \text{CoInd}\{\text{node} : \text{Nat} \rightarrow (\text{Nat} \rightarrow \text{Tree}) \rightarrow \text{Tree}\}
\end{align*}
\]

The interpretation $[\text{BTree}]$ consists of all potentially infinite terms of the form

\[\text{bnode} \, n_{1,1} \, (\text{bnode} \, n_{2,1} \, (\ldots) \, (\ldots)) \, (\text{bnode} \, n_{2,2} \, (\ldots) \, (\ldots))\]

where $n_{1,1}, n_{2,1}, n_{2,2}, \ldots \in [\text{Nat}]$. The interpretation $[\text{FTree}]$ consists of all potentially infinite terms of the form

\[\text{fnode} \, n_{1,1} \, ((\text{fnode} \, n_{2,1} \, (\ldots)) \, (\ldots)) \, (\text{fnode} \, n_{2,2} \, (\ldots) \, (\ldots)) \, \ldots : \, \text{nil}\]

where $n_{1,1}, n_{2,1}, n_{2,2}, \ldots \in [\text{Nat}]$. Finally, $[\text{Tree}]$ consists of all terms of the form $\text{node} \, n \, f$ where $n \in [\text{Nat}]$ for every $m \in [\text{Nat}]$ there is $t \in [\text{Tree}]$ such that $fm \rightarrow^\infty t$.

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

\[
\begin{align*}
\text{SPi}(B) & \quad= \quad \text{Ind}\{\text{get} : (\text{Nat} \rightarrow \text{SPi}(B)) \rightarrow \text{SPi}(B), \text{put} : \text{Nat} \rightarrow B \rightarrow \text{SPi}(B)\} \\
\text{SP} & \quad= \quad \text{CoInd}\{\text{out} : \text{SPi}(\text{SP}) \rightarrow \text{SP}\}
\end{align*}
\]

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

\[\text{odd} = \text{out}(\lambda x.\text{get}(\lambda y.\text{put}(\lambda x.\text{odd})))\]
The stream processor \texttt{odd} drops every second element of a stream, e.g., it transforms the stream $1 : 2 : 3 : 4 : \ldots$ into $1 : 3 : 5 : \ldots$. But e.g. the infinite term

\[
\text{out}(\text{get}(\lambda x_1. \text{get}(\lambda x_2. \text{get}(\lambda x_3. \text{get}(\ldots)))))
\]

is not in $[\text{SP}]$, because it nests infinitely many \texttt{gets}.

**Lemma 4.5.** If $v(i) = v′(i)$ for every $i \in \text{FSV}(\tau)$ then $[\tau]_{\xi,v} = [\tau]_{\xi,v′}$. Moreover, $[d]_{\xi,v}^{\tau} = [d]_{\xi,v′}^{\tau}$ for any $v, v′$.

*Proof.* Follows by induction on $\tau$, using the fact $\text{FSV}(\sigma_k^i) = \emptyset$ for $\sigma_k^i$ a constructor argument type as in Definition 3.3. \hfill $\Box$

**Lemma 4.6.**

1. If $\xi(\Lambda) = \xi′(\Lambda)$ for $\Lambda \in \text{TV}(\tau)$ then $[\tau]_{\xi,v}^{\tau} = [\tau]_{\xi′,v}^{\tau}$.  
2. If $\xi(B_i) = \xi′(B_i)$ for each parameter type variable $B_i$ of $\tau$, then $[d]_{\xi,v}^{\tau} = [d]_{\xi′,v}^{\tau}$.

*Proof.* Induction on $\tau$, generalizing over $\xi, \xi′$ and $v$. \hfill $\Box$

**Corollary 4.7.** If $\xi(B_i) = \xi′(B_i)$ for each parameter type variable $B_i$ of $\tau$, 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 $\tau$ is strictly positive then $[\tau]_{\xi,v}^{\tau} \subseteq [\tau]_{\xi′,v}^{\tau}$.
2. If $d$ is a (co)inductive definition then $[d]_{\xi,v}^{\tau} \subseteq [d]_{\xi′,v}^{\tau}$.
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 $\tau$, generalizing over $\xi, \xi′, v$. \hfill $\Box$

From the third point in the above lemma it follows that $[d_{\mu}]_{\xi,v}^{\mu_1} \subseteq [d_{\mu}]_{\xi,v}^{\mu_2}$ for $\mu_2 \leq \mu_1$, and $[d_{\mu}]_{\xi,v}^{\mu_1} \subseteq [d_{\mu}]_{\xi,v}^{\mu_2}$ for $\mu_1 \leq \mu_2$. 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 $d_{\mu}$, the least fixpoint of $\Phi_{d_{\mu},\xi,v}$ is then $[d_{\mu}]_{\xi,v}^{\infty}$, by how we defined $\infty$. Analogously, for a coinductive definition $d_{\nu}$, the greatest fixpoint of $\Phi_{d_{\nu},\xi,v}$ is $[d_{\nu}]_{\xi,v}^{\infty}$. Note that for $\infty \geq \infty$ we have $[d]_{\xi,v}^{\tau} = [d]_{\xi,v}^{\infty}$.

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 $t \sim t′$ then $t′ \in X$,
2. if $t \in X$ and $t \rightarrow t′$ then $t′ \in X$.

A type variable valuation $\xi$ 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 $\tau, \rho$ are strictly positive.

1. If $\xi$ is stable then so is $[\tau]_{\xi,v}^{\tau}$.
2. If $\xi$ is stable then so is $[\rho]_{\xi,v}^{\tau}$.
3. If $\xi$ and $X \subseteq \mathbb{T}^\infty$ are stable then so is $\Phi_{d_{\mu},\xi,v}(X)$. 

Proof. We show the first point by induction on \( \tau \), generalizing over \( \xi, \nu \). The remaining two points will follow directly from this proof.

First assume \( \tau = \rho^a \) with \( \rho = d(\bar{a}) \). Then \( \left[ \tau \right]_{\xi, \nu} = [\rho]_{\xi, \nu}^{v(s)} = [d]_{\xi, \nu}^{v(s)} \) where \( Y_j = \left[ \alpha_j \right]_{\xi, \nu} \) and each \( \alpha_j \) is strictly positive. By the inductive hypothesis each \( Y_j \) is stable. Hence \( \xi_1 = \xi_{[\bar{Y}/\bar{B}]} \) is also stable. We show that if \( X \subseteq T^\infty \) is stable then so is \( \Phi_d \xi_{[X/A], \nu}(X) \).

Let \( t \in \Phi_d \xi_{[X/A], \nu}(X) \). Then \( t = c t_1 \ldots t_n \) where \( t_k \in \left[ \sigma_k \right]_{\xi_{[X/A], \nu}} \) and \( c \in \text{Constr}(\rho) \) and \( \text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_n) \). Note that \( \xi_1[X/A] \) is stable, because \( X \) is. Hence \( \left[ \sigma_k \right]_{\xi_{1}[X/A], \nu} \) is stable by the inductive hypothesis.

1. Assume \( t \sim \eta t' \). Then \( t' = c t_1 \ldots t_n \) with \( t_k \sim \eta t_k' \). We have \( t_k' \in \left[ \sigma_k \right]_{\xi_{1}[X/A], \nu} \) because \( \left[ \sigma_k \right]_{\xi_{1}[X/A], \nu} \) is stable. Thus \( t' \in \Phi_d \xi_{1}[X/A], \nu(X) \).
2. Assume \( t \to^\infty t' \). Then \( t' = c t_1 \ldots t_n \) with \( t_k \to^\infty t_k' \). We have \( t_k' \in \left[ \sigma_k \right]_{\xi_{1}[X/A], \nu} \) because \( \left[ \sigma_k \right]_{\xi_{1}[X/A], \nu} \) is stable. Thus \( t' \in \Phi_d \xi_{1}[X/A], \nu(X) \).

If \( \tau = A \) is a type variable then \( \left[ \tau \right]_{\xi, \nu} = \xi(A) \) is stable because \( \xi \) is.

Assume \( \tau = \forall i \tau' \). Let \( t \in \left[ \tau \right]_{\xi, \nu} \).

1. Assume \( t \sim \eta t' \). Let \( x \in \Omega \). There is \( t_0 \) with \( t \to^\infty t_0 \in \left[ \tau' \right]_{\xi, \nu(x)} \). By Lemma 2.11 there is \( t_0' \) with \( t_0 \sim \eta t_0' \) and \( t' \to^\infty t_0' \). By the inductive hypothesis \( \left[ \tau' \right]_{\xi, \nu(x)} \) is stable, so \( t_0' \in \left[ \tau' \right]_{\xi, \nu(x)} \). Thus \( t' \in \left[ \tau \right]_{\xi, \nu} \). By confluence modulo \( \eta \) there are \( t_1, t_2 \) with \( t \to^\infty t_1 \sim \eta t_2 \) and \( t' \to^\infty t_2 \). By the inductive hypothesis \( \left[ \tau' \right]_{\xi, \nu(x)} \) is stable, so \( t_2 \in \left[ \tau' \right]_{\xi, \nu(x)} \). Thus \( t' \in \left[ \tau \right]_{\xi, \nu} \).

Finally, assume \( \tau = \xi_1 \to t_2 \) with \( t_1 \) closed and \( t_2 \) strictly positive. Let \( t \in \left[ \tau \right]_{\xi, \nu} \).

1. Assume \( t \sim \eta t' \). We need to show \( t' \in \left[ \tau \right]_{\xi, \nu} \). Let \( r \in \left[ \tau_1 \right]_{\xi, \nu} \). Then \( tr \to^\infty t_0 \in \left[ \tau_2 \right]_{\xi, \nu} \).

2. Assume \( t \to^\infty t' \). We need to show \( t' \in \left[ \tau \right]_{\xi, \nu} \). Let \( r \in \left[ \tau_1 \right]_{\xi, \nu} \). Then \( tr \to^\infty t_0 \in \left[ \tau_2 \right]_{\xi, \nu} \).

5. APPROXIMATION THEOREM

In this section we prove the approximation theorem: if \( t \to^\infty t_n \in \left[ \nu \right]_{\nu}^n \) for \( n \in \mathbb{N} \) then there exists \( t_\infty \in \left[ \nu \right]_{\nu}^\infty \) such that \( t \to^\infty t_\infty \).

The approximation theorem is an easy consequence of the following result: if \( t_n \to^\infty t_{n+1} \) and \( t_n \in \left[ \nu \right]_{\nu}^n \) for \( n \in \mathbb{N} \), then there exists \( t_\infty \) such that \( t_0 \to^\infty t_\infty \in \left[ \nu \right]_{\nu}^\infty \). If \( \nu \) is a simple coinductive type, e.g., it is a stream with a single constructor \( c \) where \( \text{ArgTypes}(c) = (\sigma, A) \), the type \( \sigma \) is closed, and \( A \) is the recursive type variable of \( \nu \), then the argument is not complicated. It follows from the assumption that \( t_{n+1} = cu_{n+1}w_{n+1} \) with \( u_{n+1} \in \left[ \sigma \right]_{\nu}^n \) and \( w_{n+1} \to^\infty w_{n+2} \). We coinductively construct \( w_\infty \) with \( w_1 \to 2^\infty w_\infty \in \left[ \nu \right]_{\nu}^\infty \) (note that \( \left[ \nu \right]_{\nu}^\infty \) treated as a unary relation may be defined coinductively). Take \( t_\infty = cu_1w_\infty \). We have \( t_0 \to 2^\infty t_\infty \in \left[ \nu \right]_{\nu}^\infty \), 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 $[-]_{\xi,v}$. We construct the term $t_\infty$ by coinduction, and show $t_0 \rightarrow^\infty t_\infty$ by coinduction, and then show $t_\infty \in [\nu]^\infty$ 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 $\tau$ be a strictly positive type and $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ a family of type variable valuations. A $\tau,\Xi$-sequence (with $v$) is a sequence of terms $\{t_n\}_{n \in \mathbb{N}}$ satisfying $t_n \in [\tau]_{\xi_n,v}$ 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 $\xi_n(A) = [\nu]^n_A$ for all $A$ and $n \in \mathbb{N}$. We usually write $\Xi_v$ instead of $\Xi_v$ when $v$ is irrelevant or clear from the context. If $T = \{\tau_A\}_{A \in \mathcal{V}}$ is a family of strictly positive types and $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ a family of type variable valuations, then $[T]_v$ denotes the family $\{\xi_n\}_{n \in \mathbb{N}}$ where $\xi_n(A) = [\tau_A]_{\xi_n,v}$. Again, the subscript $v$ is usually omitted.

A family $\Xi$ of type variable valuations is $\nu$-hereditary (with $v$) if $\Xi = \Xi_v$ or, inductively, $\Xi = \Xi'[T]$, for some $\nu$-hereditary $\Xi'$ and a family $T$ of strictly positive types.

A heredity derivation $D$ is either $\emptyset$, or, inductively, a pair $(D',T)$ where $D'$ is a heredity derivation and $T$ a family of strictly positive types. The $\nu$-hereditary family $\Xi^D$ determined by a heredity derivation $D$ is defined inductively: $\Xi^\emptyset = \Xi_v$ and $\Xi^{(D,T)} = \Xi^D[T]$.

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

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

**Lemma 5.2.** Any $\nu$-hereditary family $\Xi$ is stable.

*Proof.* By induction on the definition of a $\nu$-hereditary family, using Lemma 4.10. □

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

*Proof.* Induction on $D$. If $D = \emptyset$ then $\Xi = \Xi_v = \Xi_{v[x/i]}$ by Lemma 4.5, because $i$ does not occur in $\nu$. If $D = (D',T)$ and $\Xi = \Xi^{D'[T]}$, then by the inductive hypothesis $\Xi^{D'}$ is $\nu$-hereditary with $v[x/i]$ and determined by the heredity derivation $D'$. Assuming $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ and $\Xi^{D'} = \{\xi_n'\}_{n \in \mathbb{N}}$, we have $\xi_n(A) = [\tau_A]_{\xi_n',v} = [\tau_A]_{\xi_n,v[x/i]}$ by Lemma 4.5 because $i \notin \text{FSV}(\tau_A)$. So $\Xi$ is $\nu$-hereditary with $v[x/i]$ and determined by $D$. □

**Lemma 5.4.** If $\{t_n\}_{n \in \mathbb{N}}$ is a $A,\Xi^v$-sequence, then $t_{n+1} = c t_{n+1}^{l_1} \cdots t_{n+1}^{l_m}$ for $n \in \mathbb{N}$, and $\{t_{n+1}^k\}_{n \in \mathbb{N}}$ is a $\sigma_k,\Xi^v$-sequence for each $k = 1,\ldots,m$ where $c \in \text{Constr}(\nu)$ and $\text{ArgTypes}(c) = (\sigma_1,\ldots,\sigma_m)$ and $v = d_v(\sigma)$ and $\Xi^v = \Xi^v[T]$ where $T = \{\tau_{A'}\}_{A' \in \mathcal{V}}$ and $\tau_{B_j} = \sigma_j$ and $\tau_{A'} = A'$ for $A' \notin \{B_1,\ldots,B_l\}$ and $B_1,\ldots,B_l$ are the parameter type variables of $d_v$. 
Proof. Let \( \Xi' = \{ \xi'_n \}_{n \in \mathbb{N}} \). We have

\[
\begin{align*}
t_{n+1} &\in [A]_{\xi_{n+1},v} \\
&= [\nu]^{n+1}_{\xi_{n+1},v} \\
&= [d]^{n+1}_{\xi_{n+1},v} \\
&= \Phi_{d} (\xi_{n+1},v) \\
&= \Phi_{d}[d]^{n+1}_{\xi_{n+1},v}
\end{align*}
\]

where \( \xi(B_j) = [\alpha_j]_v \) and \( B_1, \ldots, B_l \) are the parameter type variables of \( d_v \). Then \( t_{n+1} = c_{n+1} t_{n+1}^{m_{n+1}} \) with \( t_{n+1} \in [\sigma^{n+1}] \) where \( \text{ArgTypes}(c_{n+1}) = (\sigma^1, \ldots, \sigma^{m_{n+1}}) \) and \( A \) is the recursive type variable of \( d_v \). Since \( t_n \to t_{n+1} \) for \( n \in \mathbb{N} \) we must have \( c_{n+1} = c \) and \( m_{n+1} = m \) and \( \sigma^{n+1} = \sigma_k \) for fixed \( c, m, \sigma_k \) not depending on \( n \). Also \( t_{n+1} \to t_{n+2} \) for \( k = 1, \ldots, m \) and \( n \in \mathbb{N} \). Because \( [\nu]_{\xi_{n+1},v} \) and \( \xi'_n \) are identical on \( \{ A, B_1, \ldots, B_l \} \), by Lemma 4.6 we have \( t^{n+1}_{k} \in [\sigma_k]_{\xi_{n+1},v} \). Thus \( \{ t^{k}_{n+1} \}_{n \in \mathbb{N}} \) is a \( \sigma_k, \Xi' \)-sequence.

Lemma 5.5. If \( \tau = d^{\infty}(\alpha) \) and \( \{ t_n \}_{n \in \mathbb{N}} \) is a \( \tau, \Xi \)-sequence, then \( t_n = ct^{1}_{1} \cdots t^{m}_{m} \) and \( \{ t^{k}_{n} \}_{n \in \mathbb{N}} \) is a \( \sigma_k, \Xi' \)-sequence for each \( k = 1, \ldots, m \). Then define \( f^\tau(\tau, \Xi, \{ t_n \}_{n \in \mathbb{N}}) = cr_1 \cdots r_m \) where \( r_k = f^\nu(\sigma_k, \Xi', \{ t^{k}_{n} \}_{n \in \mathbb{N}}) \).

We usually denote \( f^\tau(\tau, \Xi, \{ t_n \}_{n \in \mathbb{N}}) \) by \( t_0 \) when \( \tau, \Xi \) and \( \{ t_n \}_{n \in \mathbb{N}} \) are clear from the context.

Lemma 5.7. If \( \Xi \) is \( \nu \)-hereditary and \( \{ t_n \}_{n \in \mathbb{N}} \) is a \( \tau, \Xi \)-sequence then \( t_0 \to t_\infty \).

Proof. By Lemma 2.6 it suffices to show \( t_0 \to t_\infty \). We proceed by coinduction. By the definition of \( t_\infty \) there are the following possibilities.

- If \( \tau \) is closed then \( t_\infty = t_0 \) so \( t_0 \to t_\infty \).
- If \( \tau = A \) then without loss of generality \( \Xi = \Xi' \) and for \( n \in \mathbb{N} \) we have \( t_{n+1} = ct^{1}_{1} \cdots t^{m}_{m} \) and \( \{ t^{k}_{n+1} \}_{n \in \mathbb{N}} \) is a \( \sigma_k, \Xi' \)-sequence for \( k = 1, \ldots, m \). Then \( t_\infty = cr_1 \cdots r_m \) with \( r_k = f^\nu(\sigma_k, \Xi', \{ t^{k}_{n+1} \}_{n \in \mathbb{N}}) \). By the coinductive hypothesis \( t^1_1 \to t^2_2 \). Because \( t_0 \to t^1_{1} \), we have \( t_0 \to t_\infty \).
• If \( \tau = d^\infty(\alpha) \) then \( t_n = \alpha_1 \ldots t_m \) and \( \{t_n\}_{n \in \mathbb{N}} \) is a \( \sigma_k, \mathcal{Z}' \)-sequence for each \( k = 1, \ldots, m \). Then \( t_{\infty} = \sigma_1 \ldots r_m \) where \( r_k = f^r(\sigma_k, \mathcal{Z}', \{t_n\}_{n \in \mathbb{N}}) \). By the coinductive hypothesis \( t_k^0 \rightarrow 2^\infty r_k \), so \( t_0 \rightarrow 2^\infty t_{\infty} \).

• If \( \tau = \forall \tau' \) or \( \tau = t_1 \rightarrow t_2 \) then \( t_{\infty} = t_0 \), so \( t_0 \rightarrow 2^\infty t_{\infty} \).

We want to show that if \( \Xi \) is \( \nu \)-hereditary and \( \{t_n\}_{n \in \mathbb{N}} \) is a \( \tau, \mathcal{Z} \)-sequence, then \( t_{\infty} \in \bigcap_{n \in \mathbb{N}} \tau \xi_n \) (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 \( \tau, \Xi \)-sequence, then \( \{t_n\}_{n \in \mathbb{N}} \) is also a \( \tau, \Xi' \)-sequence.

**Proof.** Follows from definitions and Lemma 4.8.

**Lemma 5.10.** If \( t \rightarrow t' \in \llbracket \tau \rrbracket_{\xi_n, v} \) and \( \xi_n \) is stable for \( n \in \mathbb{N} \) then there exists a sequence of terms \( \{t'_n\}_{n \in \mathbb{N}} \) such that \( t \rightarrow t'_0 = w_0 = t_0 \). 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 \( t_{n+1} \rightarrow w_{n+1} \sim_{\mathcal{U}} w'_{n+1} \) and \( w_n \rightarrow w'_{n+1} \). By Lemma 2.11 there is \( t'_{n+1} \) with \( t'_n \rightarrow t'_{n+1} \) 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 \( \xi_{n+1} \) 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 \( t_{n+1} \rightarrow w_{n+1} \sim_{\mathcal{U}} t'_{n+1} \) we obtain \( t'_{n+1} \in \llbracket \tau_{n+1} \rrbracket_{\xi_{n+1}, v} \). □

**Definition 5.11.** A \( \nu \)-hereditary \( \Xi = \{\xi_n\}_{n \in \mathbb{N}} \) is **semi-complete** with \( Z, \iota \) 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 \( A, \Xi \)-sequence by Lemma 5.9) we have \( t_{\infty} = f^\nu(A, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \iota(A) \). The family \( \Xi \) is **complete** if it is semi-complete with \( \Xi, \xi_m \) for each \( m \in \mathbb{N} \).

![Figure 1: Proof of Lemma 5.10.](image)
Remark 5.12. Note that the definition of “semi-complete” depends on the implicit size variable valuation $v$, through $\Xi$ and the function $f^\nu$. Let $\Xi$ be $\nu$-hereditary (with $v$) and semi-complete with $Z, i$, with the implicit valuation $v$. Let $i$ be a fresh size variable. Then by Lemma 5.3 the family $\Xi$ is $\nu$-hereditary with $v[\nu/i]$ and determined by the same heredity derivation. It is also semi-complete with $Z, i$, with the implicit valuation $v[\nu/i]$. This is because if $\Xi$ is $\nu$-hereditary with $v[\nu/i]$ and $\{t_n\}_{n \in \mathbb{N}}$ a $\tau, \Xi$-sequence with $v[\nu/i]$, then it follows from Definition 5.6 and the statements of Lemma 5.4 and Lemma 5.5 that only the type $\tau$, 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 $\tau, \Xi$-sequence, then $t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \bigcap_{n \in \mathbb{N}}[\tau]_{\xi_n, v}$ (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 $\nu$-hereditary with $v$ and semi-complete with $Z, i$, and $\{t_n\}_{n \in \mathbb{N}}$ is a $\tau, Z$-sequence (and thus a $\tau, \Xi$-sequence by Lemma 5.9), then:

$$t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in \bigcap_{n \in \mathbb{N}}[\tau]_{\xi_n, v}.$$

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

We are now going to show that every $\nu$-hereditary family $\Xi$ is complete. To achieve this we show that $\Xi^\nu$ is complete (Corollary 5.16), and that if $\Xi$ is complete then so is $\Xi[7]$ (Lemma 5.17).

Lemma 5.15. If $\Xi$ is semi-complete with $Z, i$ then $\Xi[7]$ is semi-complete with $Z[7]$, i.e., where $\mathcal{T} = \{\tau_A\}_{A \in V_T}$ and $\iota'(A) = [\tau_A]_{\iota, v}$.

Proof. Let $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ and $\Xi = \Xi[7] = \{\xi'_n\}_{n \in \mathbb{N}}$ and $Z = \{\zeta_n\}_{n \in \mathbb{N}}$ and $Z' = Z[7] = \{\zeta'_n\}_{n \in \mathbb{N}}$. We have $\zeta'_n(A) = [\tau_A]_{\iota, v} \subseteq [\tau_A]_{\iota, \zeta_n, v} = \zeta_n(A)$ 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., $t_n \in \mathcal{T}$ for $n \in \mathbb{N}$. Then $\{t_n\}_{n \in \mathbb{N}}$ is also a $A, Z$-sequence. Because $\Xi$ is semi-complete with $Z, i$, by Lemma 5.13 we have $t_\infty = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) = f^\nu(\tau, \Xi, \{t_n\}_{n \in \mathbb{N}}) \in [\tau]_{\xi_n, v}$.

Corollary 5.16. If $\Xi$ is complete then so is $\Xi[7]$.

Lemma 5.17. $\Xi^\nu$ is complete.

Proof. We show by induction on $m \in \mathbb{N}$ that $\Xi^\nu$ is semi-complete with $\Xi, \xi_m^\nu$. We have $\Xi^\nu \subseteq \Xi^\nu$. Also $\Xi^\nu$ is stable. Let $\{t_n\}_{n \in \mathbb{N}}$ be a $A, \Xi^\nu$-sequence. We need to show $t_\infty = f^\nu(\tau, \Xi', \{t_n\}_{n \in \mathbb{N}}) \in [\nu]_{\iota, v}^m(A)$. If $m = 0$ then $[\nu]_{\iota, v}^m = \mathcal{T}$, so $t_\infty \in [\nu]_{\iota, v}^m(A)$. Assume $m = m' + 1$. We have $t_n \in [\xi_m(A)]_{\nu, v}$, and $t_\infty = f^\nu(\tau, \Xi', \{t_n\}_{n \in \mathbb{N}}) \in [\tau]_{\iota, v}^m(A)$. Then by Lemma 5.4 we have $t_{n+1} = \cdots$ for $n \in \mathbb{N}$ and $\{t_{n+1}\}_{n \in \mathbb{N}}$ is a $\sigma, i$-sequence for each $i = 1, \ldots, k$ where $c \in \text{Constr}(\nu)$ and $\text{ArgTypes}(c) = \{\sigma_1, \ldots, \sigma_k\}$ and $\nu = d_m(\alpha)$ and $\Xi^\nu = \Xi'[7]$ where $\mathcal{T} = \{\tau_A\}_{A \in V_T}$ and $\tau_B_j = \alpha_j$ and $\tau_A = A$ for $A \notin \{B_1, \ldots, B_l\}$ and $B_1, \ldots, B_l$ are the parameter type variables of $d_m$. By the inductive hypothesis $\Xi^\nu$ is semi-complete with $\Xi', \xi_m^{\nu'}$. By Lemma 5.15 we conclude that $\Xi'$ is semi-complete with $\Xi', i$ where $\iota(A) = [\tau_A]_{\xi_m^{\nu'}, v}$, i.e.,
Appendix A. We treat three cases that differ more substantially.

Follows by induction from Lemma 5.17 and Corollary 5.16.

Proof. ⋂

fixpoint for any coinductive type. For this we need the following lemma about intersection ⋂

Corollary 5.19. Every ν-hereditary family is complete.

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

Corollary 5.18. If Ξ = {ξn}n∈N is ν-hereditary and {tn}n∈N is a τ,Ξ-sequence, then t∞ = fν(τ, Ξ; {tn}n∈N) ∈ ⋂n∈N[τ]ξn,v.

Proof. Follows from Corollary 5.14 and Corollary 5.18.

We are now going to show that [ν] v A = [ν] ∞ A, 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 ⋂n∈N ξn by (⋂n∈N ξn)(A) = ⋂n∈N ξn(A) for any A.

Lemma 5.20. If Ξ = {ξn}n∈N is complete then ⋂n∈N[τ]ξn,v ⊆ [τ] ⋂n∈N ξn,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 τ = dµ(α) then ⋂n∈N[τ]ξn,v = ⋂n∈N[µ][ξv]n,v, where µ = dµ(α). By induction on x we show ⋂n∈N[µ][ξv]n,v ⊆ ⋂n∈N[µ][ξv]n,v. There are three cases.

1. x = 0. Then ⋂n∈N[µ][ξv]n,v = ⋂n∈N 0 = 0 = [µ][ξv] ⋂n∈N ξn,v.

2. x = x' + 1. Let τ ∈ ⋂n∈N[µ][ξv]n,v. Then τ = cu1…uk with ui ∈ ⋂n∈N[σi]ξn,v where A is the recursive type variable of dµ and c ∈ Constr(dµ) and ArgTypes(c) = (σ1,…,σk) and B1,…,Bt are the parameter type variables of dµ and Ξ = Ξ[T] and Ξ' = {ξn}n∈N and T = {τA}A∈VT and τBj = αj and τA = µi and τA = A' for A' ≠ {A,B1,…,Bt} where i is a fresh size variable such that v(i) = x' (by Lemma 5.4 we may assume τBj is a size variable exists). So Ξ' is also complete by Corollary 5.16. By the main inductive hypothesis we show ⋂n∈N[µ][ξv]n,v for any strictly positive τ.

We have ξ(A) = ⋂n∈N[µ][ξv]n,v ⊆ [µ][ξv] ⋂n∈N ξn,v by the inductive hypothesis. Also ⋂n∈N[ξn](Bj) = ⋂n∈N[ξn](Bj) = ⋂n∈N[αj][ξv]n,v ⊆ [αj] ⋂n∈N ξn,v by the inductive hypothesis and Lemma 4.6, because we may assume B1,…,Bt ∈ TV(αj).

Hence t = ⋂n∈N[µ][ξv]n,v.

Therefore t ∈ ⋂n∈N[µ][ξv]n,v.

3. x is a limit ordinal. Let t ∈ ⋂n∈N[µ][ξv]n,v = ⋂n∈N ⋃n< x [µ][ξv]n,v. Then for each n ∈ N there is x n < x with t ∈ [µ][ξv]n,v, i.e., t ∈ ⋂n∈N[µ][ξv]n,v. We have x n > 0 is a successor ordinal for n ∈ N, because [µ][ξv] 0 n,v = 0. Because Ξ is stable by Lemma 5.2, using Lemma A.2 we conclude t ∈ ⋂n∈N[µ][ξv]n,v. Then t ∈ [µ][ξv]n,v by an argument as in the previous point.
Theorem 5.22

We have \( \xi \) \( \text{Approximation Theorem} \)

Theorem 5.22

Proof.

By Lemma 5.2 and Lemma 5.10 there exists a sequence of terms \( \{t'_n\}_{n \in \mathbb{N}} \) such that \( t \xrightarrow{\infty} t_0 \) and \( t'_n \in [\tau]_{\xi_n,\nu;[\xi'|i]} \) and \( t'_n \xrightarrow{\infty} t'_{n+1} \) for \( n \in \mathbb{N} \). Thus \( \{t'_n\}_{n \in \mathbb{N}} \) is a \( \tau',\Xi\)-sequence (with \( v[\xi'|i]\)). Because \( \Xi \) is complete, by Corollary 5.14 there is \( t^\omega \) with \( t \xrightarrow{\infty} t^\omega \in \bigcap_{n \in \mathbb{N}} [\tau']_{\xi_n,\nu;[\xi'|i]} \). By the inductive hypothesis \( t^\omega \in [\tau']_{\xi,\nu;[\xi'|i]} \). Since \( \omega \in \Omega \), this implies \( t \in [\tau']_{\xi,\nu;[\xi'|i]} \).

If \( \tau = \forall i.\tau' \) then let \( t \in \bigcap_{n \in \mathbb{N}} [\tau]_{\xi_n,\nu} \). Let \( \omega \in \Omega \). For \( n \in \mathbb{N} \) there is \( t_n \) with \( t \xrightarrow{\infty} t_n \in [\tau']_{\xi_n,\nu;[\xi'|i]} \). By Lemma 5.2 and Lemma 5.10 there exists a sequence of terms \( \{t'_n\}_{n \in \mathbb{N}} \) such that \( t \xrightarrow{\infty} t_0 \) and \( t'_n \in [\tau']_{\xi_n,\nu;[\xi'|i]} \) and \( t'_n \xrightarrow{\infty} t'_{n+1} \) for \( n \in \mathbb{N} \). Thus \( \{t'_n\}_{n \in \mathbb{N}} \) is a \( \tau',\Xi\)-sequence (with \( v[\xi'|i]\)). Because \( \Xi \) is complete, by Corollary 5.14 there is \( t^\omega \) with \( t \xrightarrow{\infty} t^\omega \in \bigcap_{n \in \mathbb{N}} [\tau']_{\xi_n,\nu;[\xi'|i]} \). By the inductive hypothesis \( t^\omega \in [\tau']_{\xi,\nu;[\xi'|i]} \). Since \( \omega \in \Omega \), this implies \( t \in [\tau']_{\xi,\nu;[\xi'|i]} \).

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

Lemma 5.21. \( [\nu^\omega]_v = [\nu]_v^\infty \).

Proof. It suffices to show \( [\nu^\omega]_v \subseteq [\nu]_v^\infty \). So let \( t \in [\nu]_v^\infty \). Then \( t \in [\nu]_v^m \) for each \( m \in \mathbb{N} \). So \( t = cu_1 \ldots u_k \) where \( u_i \in [\sigma]_{\xi'_m,\nu} \) for \( m \in \mathbb{N} \) where \( \Xi' = \{\xi'_m\}_{m \in \mathbb{N}} \) and \( \Xi = \Xi' [\tau] \) and \( \tau = \{\tau_A\}_{A \in V_T} \) and \( \tau_B = \{\alpha_j\}_j \) and \( \tau_A = A \) for \( A \notin \{B_1, \ldots, B_i\} \) and \( \nu = d(\bar{\alpha}) \) and \( B_1, \ldots, B_i \) are the parameter type variables of \( d\). Note that \( \Xi' \) is complete by Corollary 5.18. Hence by Lemma 5.20 we have \( u_i \in [\sigma]_{\xi'_m,\nu} \). Let \( \xi' = \bigcap_{m \in \mathbb{N}} \xi'_m \). We have \( \xi'(A) = \bigcap_{m \in \mathbb{N}} \xi_m(A) = \bigcap_{m \in \mathbb{N}} \xi'_m(A) = \bigcap_{m \in \mathbb{N}} [\nu]_v^m = [\nu]_v^\infty \) where \( A \) is the recursive type variable of \( d\nu \), and \( \xi'(B_i) = \{\alpha_j\}_j \). Therefore \( t \in [\nu]_v^\infty \). \( \square \)

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 \( t \xrightarrow{\infty} t_n \in [\nu]_v^n \) for \( n \in \mathbb{N} \) then there exists \( t_\infty \in [\nu]_v^\infty \) such that \( t \xrightarrow{\infty} t_\infty \).

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

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 \( t \in T^\infty \) and let \( f : \mathbb{N} \to \mathbb{N} \) be such that \( \lim_{n \to \infty} f(n) = \infty \). Assume that for every \( n \in \mathbb{N} \) and every \( w_n \in [\nu_1]^n \) there is \( w_n \) with \( tu_n \to \infty w_n \in [\nu_2]^{f(n)} \). Then \( t \in [\nu_1 \to \nu_2] \), i.e., for every \( u \in [\nu_1] \) there is \( w \) with \( tu \to \infty w \in [\nu_2] \).

Proof. Let \( u \in [\nu_1] \). Because \([\nu_1] = [\nu_1]^\infty \subseteq [\nu_1]^n \), for each \( n \in \mathbb{N} \) there is \( w_n \) with \( tu \to \infty w_n \in [\nu_2]^{f(n)} \). Because \( \lim_{n \to \infty} f(n) = \infty \), 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 \( [\nu_2]^{f(n_k)} \subseteq [\nu_2]^k \). This implies that for each \( k \in \mathbb{N} \) there is \( w_{n_k} \) with \( tu \to \infty w_{n_k} \in [\nu_2]^k \).

Now by Theorem 5.22 there is \( w \) with \( tu \to \infty w \in [\nu_2]^\infty \).

The above result is, however, a bit unsatisfying in that the valuation approximations \([\nu_1]^n \) contain too many terms, i.e., they contain all terms which nest at least \( n \) constructors of the coinductive type \( \nu_1 \). 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 \( \bot = (\lambda x.xx)(\lambda x.xx) \). Note that \( \bot \) is the only redcut of \( \bot \).

For a coinductive definition \( d_\nu \) and \( n \in \mathbb{N} \) we define the strict valuation approximation \([d_\nu]^n_\bot \subseteq T^\infty \) as follows: \([d_\nu]^0_\bot = \{\bot\} \), \([d_\nu]^{n+1}_\bot = \Phi d_\nu, \xi, v ([d_\nu]^n_\bot, \xi, v) \). We set \([\nu]^{n}_\bot = [d_\nu]^n_\bot \{\nu_\xi, v \} \) where \( \nu = d_\nu(\bar{a}) \) is a coinductive type, \( Y_j = [\nu_\xi, v] \), and \( \bar{B} \) are the parameter type variables of \( d_\nu \).

The relation \( \triangleright \) is defined coinductively.

\[
\begin{align*}
\frac{}{t \triangleright \bot} & \quad \frac{}{x \triangleright x} & \quad \frac{}{c \triangleright c} \\
\frac{t \triangleright t'}{\lambda x.t \triangleright \lambda x.t'} & \quad \frac{t_1 \triangleright t'_1 \quad t_2 \triangleright t'_2}{t_1 t_2 \triangleright t'_1 t'_2} & \quad \frac{t \triangleright t'}{t_k \triangleright t'_k} \\
\end{align*}
\]

In other words, \( t \triangleright t' \) if \( t' \) is \( t \) with some subterms replaced by \( \bot \). If \( t \triangleright t' \) and \( t', t \in [\nu] \) then \( t' \in [\nu]_\bot \) and \( t' \) is an approximant of \( t \) of size \( n \).

Lemma 5.25. If \( t \triangleright t' \to u' \) then there is \( u \) with \( t \to^u u' \).

Proof. Induction on \( t' \to u' \).

Lemma 5.26. If \( t \triangleright t' \to^\infty u' \) then there is \( u \) with \( t \to^\infty u \to u' \).

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

A set \( X \subseteq T^\infty \) is approximation expansion closed if \( t' \in X \) and \( t \triangleright t' \) imply \( t \in X \).

Lemma 5.27. Assume \( \xi(A) \) is approximation expansion closed for every \( A \). Then \([\tau]_{\xi,v} \) is approximation expansion closed.

Proof. Induction on \( \tau \), using Lemma 5.26 for the cases \( \tau = \tau_1 \to \tau_2 \) and \( \tau = \forall i.t' \).
Theorem 5.28. Let \( t \in \mathbb{T}^\infty \) and let \( f : \mathbb{N} \to \mathbb{N} \) be such that \( \lim_{n \to \infty} f(n) = \infty \). Let \( u \in \nu_1 \). If for every \( n \in \mathbb{N} \) and every \( u_n \in \nu_1^f(n) \) with \( u \succ u_n \) there is \( w_n \) with \( tu_n \to w_n \in \nu_2^{f(n)} \), then there is \( w \) with \( tu \to w \in \nu_2 \).

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

6. The type system \( \lambda^\odot \)

In this section we define the type system \( \lambda^\odot \) 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 \( \tau \) in the system \( \lambda^\odot \) then its erasure infinitarily reduces to a \( t' \in \tau \).

Decorated terms are given by:
\[
\begin{align*}
t & ::= \ x \mid c \mid \lambda x : \tau . t \mid tt \mid ts \mid \Lambda i . t \mid \text{case}(t; \{c_k \mapsto t_k\}) \mid \text{fix}\ f : \tau.t \mid \text{cofix}\ j\ f : \tau.t
\end{align*}
\]
where \( x \in \mathcal{V} \), and \( c, c_k \in \mathcal{C} \), and \( \tau \) 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 \( \text{tgt} \) that gives the target of a type is defined as follows:
\[
\begin{align*}
\text{tgt}(A) & = A, \text{tgt} (\rho^s) = \rho^s, \\
\text{tgt}(\tau_1 \to \tau_2) & = \text{tgt}(\tau_2), \\
\text{tgt}(\forall i . \tau) & = \text{tgt}(\tau).
\end{align*}
\]

By \( \text{chgtgt} (\tau, \alpha) \) we denote the type \( \tau \) with the target exchanged for \( \alpha \). Formally, \( \text{chgtgt} (\tau, \alpha) \) is defined inductively:
\[
\begin{align*}
\text{chgtgt} (A, \alpha) & = \alpha, \text{tgt} (\rho^s, \alpha) = \alpha, \\
\text{chgtgt} (\tau_1 \to \tau_2, \alpha) & = \tau_1 \to \text{chgtgt} (\tau_2, \alpha), \\
\text{chgtgt} (\forall i . \tau, \alpha) & = \forall i . \text{chgtgt} (\tau, \alpha).
\end{align*}
\]

Note that free size variables in \( \alpha \) may be captured as a result of this operation.

A context \( \Gamma \) is a finite map from type variables to types. We write \( \Gamma, x : \alpha \) to denote the context \( \Gamma' \) 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 \( \lambda^\odot \) 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 \( \tau \).

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 \( \rho \) may be assigned an appropriate type with the size of the recursive occurrences of \( \rho \) being \( s \), then \( ct_1 \ldots t_n \) has type \( \rho^{s+1} \). For instance, for the type of lists of natural numbers \( \text{List}(\text{Nat}) \), the rule (con) says that if \( x : \text{Nat} \) and \( y : \text{List}'(\text{Nat}) \) then \( \text{cons}\ xy : \text{List}^{i+1}(\text{Nat}) \).
The (case) rule allows to type case expressions. If the decorated term $t$ that is matched on has a (co)inductive type $\rho^{s+1}$, and for each $k = 1, \ldots, n$ under the assumption that the arguments of the constructor $c_k$ have appropriate types (with the recursive occurrences of $\rho$ having size $s$) the branch $t_k$ may be given the type $\tau$, then the case expression has type $\tau$.

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 \( Y = (\lambda x. \lambda f. f(xx f)) (\lambda x. \lambda f. f(xx f)) \) be the Turing fixpoint combinator.

Note that \( Y t \to^* t(Y t) \) for any term \( t \).

The erasure \( |t| \) of a decorated term \( t \) is defined inductively:

\[
\begin{align*}
| x | &= x, \\
| c | &= c, \\
| \lambda x : \tau. t | &= \lambda x. |t|, \\
| t_1 t_2 | &= |t_1| |t_2|, \\
| t s | &= |t|, \\
| \text{case}(t; \{ c_k \bar{x} \Rightarrow t_k \}) | &= \text{case}(|t|; \{ c_k \bar{x} \Rightarrow |t_k| \}), \\
| \text{fix} f : \tau. t | &= Y(\lambda f. |t|), \\
| \text{cofix} f : \tau. t | &= Y(\lambda f. |t|).
\end{align*}
\]

6.1. **Examples.** In this section, we give a few examples of typing derivations in the system \( \lambda^{\mathcal{O}} \). 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 \( \lambda^{\mathcal{O}} \).

**Example 6.2.** We reuse the definitions of Nat and Strm from Example 3.4 (see also Example 4.2). Consider the function \( \text{tl} = \Lambda i. \lambda s : \text{Strm}^{i+1}. \text{case}(s; \{ \text{cons} x t \Rightarrow t \}) \)

To type \( \text{tl} \) 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 \( t : \text{Strm}^j \), so the match has type \( \text{Strm}^j \) by the (case) rule. Hence, by the (lam) and (gen) rules we obtain \( \vdash \text{tl} : \forall i. \text{Strm}^{i+1} \to \text{Strm}^i \).

Similarly, the function

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

may be assigned the type \( \forall i. \text{Strm}^{i+1} \to \text{Nat} \).

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

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

\[
\begin{align*}
\text{run} &= \text{cofixrun} : \text{SP} \to \text{Strm} \to \text{Strm}.
\end{align*}
\]

\[
\begin{align*}
\lambda x : \text{SP}. \lambda y : \text{Strm}.
\end{align*}
\]

\[
\begin{align*}
\text{case}(x; \{ \text{out} z \Rightarrow \text{runi} z y \})
\end{align*}
\]

where

\[
\begin{align*}
\text{runi} &= \text{fixruni} : \text{SP}(\text{SP}) \to \text{Strm} \to \text{Strm}^{j+1}.
\end{align*}
\]

\[
\begin{align*}
\lambda z : \text{SP}(\text{SP}). \lambda y : \text{Strm}.
\end{align*}
\]

\[
\begin{align*}
\text{case}(z; \{ \text{get} f \Rightarrow \text{runi}(f(\text{hd} \circ y))(\text{tl} \circ y), \text{put} n x' \Rightarrow n :: \text{run} x' y \})
\end{align*}
\]

Recall that Strm with no decorations is an abbreviation for \( \text{Strm}^\infty \).

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

\[
\text{run} : \text{SP} \to \text{Strm} \to \text{Strm}^j
\]

and \( x : \text{SP} \) and \( y : \text{Strm} \).

- To type \( \text{runi} \) assume \( \text{runi} : \text{SP}(\text{SP}) \to \text{Strm} \to \text{Strm}^{j+1} \) and \( z : \text{SP}(\text{SP}) \) and \( y : \text{Strm} \). We apply the (case) rule to type the match inside \( \text{runi} \). For this purpose we need to type both branches.
Assuming \( f : \text{Nat} \rightarrow \text{SPi}(\text{SP}) \), we have \( \text{runi}(f(\text{hd} \infty y))(\text{tl} \infty y) : \text{Strm}^{j+1} \) 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 \( \text{run} x' y : \text{Strm}^{j} \) by the (app) rule. Thus \( n :: \text{run} x' y : \text{Strm}^{j+1} \) by (con).

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

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

by the (fix) typing rule.

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

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

\[
\text{run} : \text{SP} \rightarrow \text{Strm} \rightarrow \text{Strm}^{j}
\]

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

**Remark 6.4.** Strictly speaking, it is possible to type non-productive terms in our system. For instance, the term \( t = \text{cofix} f : \text{Strm}^{0}.f \) has type \( \text{Strm}^{0} \). 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 \( \nu s \) 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 \( \lambda^{\text{\textdollar}} \) 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 \( \lambda^{\text{\textdollar}} \) is coNP-complete. More precisely, given \( \Gamma, t, \tau \) 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 $\Gamma \vdash t : \tau$ then there is $t' \in [\tau]$ with $|t| \to^\infty t'$. 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 $\lambda^\Diamond$ 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 $r = \Gamma(\lambda f.t)$ with $\operatorname{tgt}(\tau) = v^s$. Let $r_0 = r$ and $r_{n+1} = t[r_n/f]$ for $n \in \mathbb{N}$. Let $r' = \chi_{\lambda r_n}$ where $j \notin \operatorname{SV}(s, v, \tau)$. If for every $n \in \mathbb{N}$ there is $r'_n$ with $r_n \to^\infty r'_n \in [\tau]_{v[n/j]}$ then there is $r'$ with $r \to^\infty r' \in [\tau]_v$.

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

Without loss of generality assume $\tau = \forall i_1.\nu_1^1 \to \forall i_2.\nu_2^2 \to v^s$. Then $\chi_{\lambda r_n} \in [\nu_1^1[v_1^1/i_1], \nu_2^2[v_2^2/i_1, i_2/i_2]].$ Then because $j \notin \operatorname{SV}(s, v, \tau)$, using Lemma 4.5, we conclude that for every $n \in \mathbb{N}$ there is $r'_n$ with $r'_n u_1 u_2 \to^\infty r''_n \in [v]^\min(m, n)$ where $m = v[i_1, i_2/i_2]$. It suffices to find $r'$ with $ru_1 u_2 \to^\infty r' \in [v]^\infty$.

First assume $m < \omega$. Then $r''_n \in [v]^m$ so we may take $r' = r''_m$.

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

Lemma 7.2. If $\tau \sqsubseteq \tau'$ then $[\tau]_v \subseteq [\tau']_v$.

Proof. Induction on $\tau$. If $\tau = d^s(\alpha) \sqsubseteq \tau'$ then $d^s(\beta)$ where $s \leq s'$ and $\alpha \subseteq \beta$. We have $[\tau]_v = [d^s(\alpha)]_{v[s]} = [d^s(\beta)]_{v[s]}$. By the inductive hypothesis $[\alpha]_v \subseteq [\beta]_v$. Let $\xi \in [\beta]_v$. Then $\xi \subseteq [\tau]'_v$. By Lemma 4.8 we obtain $[d^s](\xi) \subseteq [d^s](\xi)'$. Also $v(s) \leq v(s')$ and $[\tau]'_v = [d^s(\xi)]_{v[s']}$. Thus $[\tau]'_v = [d^s(\xi)]_{v[s']} \subseteq [d^s(\beta)]_{v[s']} \subseteq [\tau]'_v$.

If $\tau = \forall i.\tau_1 \sqsubseteq \tau'$ then the argument is analogous to the previous case.

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

Theorem 7.3 (Soundness). If $\Gamma \vdash t : \tau$ with $\Gamma = \tau_1 \to \tau_2 \to \tau_3$ for every size variable valuation $v : V_S \to \Omega$ and all $t_1 \in [\tau_1]_v$, $t_2 \in [\tau_2]_v$ there exists $t'$ such that $|t|[t_1/x_1, \ldots, t_n/x_n] \to^\infty t' \in [\tau]_v$.

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.
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:

\[ t \rightarrow^\infty t_n \in \nu^n \text{ for } n \in \mathbb{N} \text{ then there exists } t_\infty \in \nu^\infty \text{ such that } t \rightarrow^\infty t_\infty, \]

where \( \nu \) is a coinductive type.

In the second part of the paper, we defined a type system \( \lambda^\diamond \) 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 \( \tau \) in the system then its erasure infinitarily reduces to a term \( t' \in [\tau] \). Together with confluence modulo \( U \) of the infinitary reduction relation and the stability of \( [\tau] \), 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 \( \lambda^\diamond \) a simple (co)inductive type \( \rho \) such that \([\rho]\) 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 \([\text{Strm}]\) are normal forms. If all elements of \([\rho]\) 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 \( t \in [\tau_1 \rightarrow \tau_2] \) is very natural and ensures the productivity of \( t \) regarded as a function: we require that for \( u \in [\tau_1] \) there is \( u' \in [\tau_2] \) 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


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 $\tau$ is strictly positive and $\xi_1, \xi_2$ are stable then $[\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} = [\tau]_{\xi_1 \cap \xi_2,v}$, where we define $(\xi_1 \cap \xi_2)(A) = \xi_1(A) \cap \xi_2(A)$ for any type variable $A$.

**Proof.** Induction on $\tau$. Note that it suffices to show $[\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} \subseteq [\tau]_{\xi_1 \cap \xi_2,v}$, 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 $\tau$ is closed then $[\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} = [\tau]_{\xi_1,v} \cap [\tau]_{\xi_1,v} = [\tau]_{\xi_1 \cap \xi_2,v}$ by Lemma 4.6.
- If $\tau = A$ then $[\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} = \xi_1(A) \cap \xi_2(A) = [\tau]_{\xi_1 \cap \xi_2,v}$.
- If $\tau = d_{\mu}^{\nu}(\bar{\alpha})$ then $[\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} = [d_{\mu}]_{\xi_1,v}^{\nu(s)} \cap [d_{\mu}]_{\xi_2,v}^{\nu(s)}$ where $\xi_n(B_j) = [\alpha_j]_{\xi_n}$ and $\xi_n(A') = \xi_n(A)$ for $A \notin \{B_1, \ldots, B_l\}$ and $B_1, \ldots, B_l$ are the parameter type variables of $d_{\mu}$. By induction on $\nu$ we show $[d_{\mu}]_{\xi_1,v}^{\nu(s)} \cap [d_{\mu}]_{\xi_2,v}^{\nu(s)} \subseteq [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu(s)}$. There are three cases.
  1. $\nu = 0$. Then $[d_{\mu}]_{\xi_1,v}^{\nu(s)} \cap [d_{\mu}]_{\xi_2,v}^{\nu(s)} = \emptyset \cap \emptyset = \emptyset = [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu(s)}$.
  2. $\nu = \nu + 1$. Let $t \in [d_{\mu}]_{\xi_1,v}^{\nu(s)} \cap [d_{\mu}]_{\xi_2,v}^{\nu(s)}$. Then $t = cu_1 \ldots u_k$ with $u_i \in \bigcap_{n \in \{1, 2\}} [\sigma_i]_{\xi_n}^{\nu(s)}[[d_{\mu}]_{\xi_n,v}^{\nu(s)}/A,v]$.

where $A$ is the recursive type variable of $d_{\mu}$ and $c \in \text{Constr}(d_{\mu})$ and $\text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_k)$. By the main inductive hypothesis $u_i \in [\sigma_i]_{\xi_n}$ where

$$\xi = \bigcup_{n \in \{1, 2\}} \xi_n([d_{\mu}]_{\xi_n,v}^{\nu(s)}/A,v).$$

We have

$$\xi(A) = \bigcap_{n \in \{1, 2\}} [d_{\mu}]_{\xi_n,v}^{\nu(s)} = [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu(s)}$$

by the inductive hypothesis. Hence

$$\xi = ([\xi_1 \cap \xi_2])[[d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu(s)}/A].$$

Therefore $t \in [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu(s)}$.

3. $\nu$ is a limit ordinal. Let

$$t \in \bigcap_{n \in \{1, 2\}} [d_{\mu}]_{\xi_n,v}^{\nu} = \bigcup_{n \in \{1, 2\}} [d_{\mu}]_{\xi_n,v}^{\nu}.$$  

Then for each $n \in \{1, 2\}$ there is $\nu_n < \nu$ with $t \in [d_{\mu}]_{\xi_n,v}^{\nu_n}$. We have $\nu_n > 0$ is a successor ordinal for $n = 1, 2$, because $[d_{\mu}]_{\xi_n,v}^{0} = \emptyset$. Thus $t = cu_1 \ldots u_k$ with $u_i \in \bigcap_{n \in \{1, 2\}} [\sigma_i]_{\xi_n}^{\nu_n}[[d_{\mu}]_{\xi_n,v}^{\nu_n-1}/A,v]$ where $A$ is the recursive type variable of $d_{\mu}$ and $c \in \text{Constr}(d_{\mu})$ and $\text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_k)$. By the main inductive hypothesis $u_i \in [\sigma_i]_{\xi_n}$ where $\xi = \bigcap_{n \in \{1, 2\}} [\sigma_i]_{\xi_n}^{\nu_n}[[d_{\mu}]_{\xi_n,v}^{\nu_n-1}/A]$. Without loss of generality assume $\nu_1 \leq \nu_2$. We have $\xi(A) = \bigcap_{n \in \{1, 2\}} [d_{\mu}]_{\xi_n,v}^{\nu_n-1} \subseteq [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu_2-1} \cap [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu_2-1}$. Hence by the inductive hypothesis $\xi(A) \subseteq [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu_2-1}$. Thus by Lemma 4.8 we have $u_i \in [\sigma_i]_{\xi_n}$ where $\xi' = ([\xi_1 \cap \xi_2])[[d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu_2-1}/A]$. Therefore $t \in [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu_2} \subseteq [d_{\mu}]_{\xi_1 \cap \xi_2,v}^{\nu}$. 


We have thus shown \([\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} \subseteq [d_{\mu}]^{\nu(s)}_{\xi_1,\xi_2,v}\). Note that \(\xi_1,\xi_2\) are stable by Lemma 4.10. We have \((\xi_1 \cap \xi_2)(B_j) = \xi_1(B_j) \cap \xi_2(B_j) = [\alpha_j]_{\xi_1,v} \cap [\alpha_j]_{\xi_2,v} = [\alpha_j]_{\xi_1,\xi_2,v}\) by the inductive hypothesis and Lemma 4.6, because we may assume \(B_1, \ldots, B_l \notin \text{TV}(\alpha_j)\). Hence \([\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} \subseteq [d_{\mu}(\alpha)]^{\nu(s)}_{\xi_1,\xi_2,v}\) by Lemma 4.6.

- If \(\tau = d_{\nu}(\alpha)\) then \([\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} = [d_{\nu}]^{\nu(s)}_{\xi_1,\xi_2,v}\) where \(\xi_1,\xi_2\) are the parameter type variables of \(d_{\nu}\). Note that \(\xi_1,\xi_2\) are stable by Lemma 4.10. First, by induction on \(\nu\) we show \([d_{\nu}]_{\xi_1,v} \cap [d_{\nu}]_{\xi_2,v} \subseteq [d_{\nu}]_{\xi_1,\xi_2,v}\). If \(\nu = 0\) then \([d_{\nu}]_{\xi_1,v} \cap [d_{\nu}]_{\xi_2,v} = \nu = [d_{\nu}]_{\xi_1,\xi_2,v}\). So assume \(\nu = \nu + 1\). Let \(t \in [d_{\nu}]_{\xi_1,v} \cap [d_{\nu}]_{\xi_2,v}\). Then \(t = cu_1 \ldots u_k\) with \(u_i \in \bigcap_{n \in \{1,2\}} [\sigma_i]_{\xi_1,\xi_2,v}/[d_{\nu}]_{\xi_1,\xi_2,v}/A\) where \(A\) is the recursive type variable of \(d_{\nu}\) and \(c \in \text{Constr}(d_{\nu})\) and \(\text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_k)\). By Lemma 4.10 and the main inductive hypothesis \(u_i \in [\sigma_i]_{\xi_1,\xi_2,v}\) where \(\xi = \bigcap_{n \in \{1,2\}} [d_{\nu}]_{\xi_1,\xi_2,v}/A\). By the inductive hypothesis

\[\xi(A) = [d_{\nu}]_{\xi_1,v} \cap [d_{\nu}]_{\xi_2,v} = [d_{\nu}]_{\xi_1,\xi_2,v}\]

Hence \(t \in [d_{\nu}]_{\xi_1,\xi_2,v}\). Finally, assume \(\nu\) is a limit ordinal. Then

\[\bigcap_{n \in \{1,2\}} [d_{\nu}]_{\xi_1,v} = \bigcap_{n \in \{1,2\}} \bigcap_{\nu' < \nu} [d_{\nu}]_{\xi_1,v}^\nu\]

\[\subseteq \bigcap_{\nu' < \nu} \bigcap_{n \in \{1,2\}} [d_{\nu}]_{\xi_1,v}^{\nu'}\]

Now like in the previous point

\((\xi_1 \cap \xi_2)(B_j) = \xi_1(B_j) \cap \xi_2(B_j) = [\alpha_j]_{\xi_1,v} \cap [\alpha_j]_{\xi_2,v} = [\alpha_j]_{\xi_1,\xi_2,v}\) by the inductive hypothesis and Lemma 4.6. Hence

\([\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} = [d_{\nu}]_{\xi_1,v}^{\nu} \cap [d_{\nu}]_{\xi_2,v}^{\nu} \subseteq [d_{\nu}]_{\xi_1,\xi_2,v}^{\nu}\]

by Lemma 4.6.

- Suppose \(\tau = \forall i.\tau'\). Let \(t \in [\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v}\) and \(\nu \in \Omega\). There are \(t_1, t_2\) with \(t \rightarrow^\infty t_1 \in [\tau]_{\xi_1,v}^{[\xi_1,\xi_2,v]}\) and \(t \rightarrow^\infty t_2 \in [\tau]_{\xi_2,v}^{[\xi_1,\xi_2,v]}\). 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 \(t'_2 \in [\tau]_{\xi_1,v}^{[\xi_1,\xi_2,v]} \cap [\tau]_{\xi_2,v}^{[\xi_1,\xi_2,v]}\). Hence \(t \rightarrow^\infty t'_2 \in [\tau]_{\xi_1,v}^{[\xi_1,\xi_2,v]} \cap [\tau]_{\xi_2,v}^{[\xi_1,\xi_2,v]}\) by the inductive hypothesis. Thus \(t \in [\tau]_{\xi_1,\xi_2,v}\). This shows \([\tau]_{\xi_1,v} \cap [\tau]_{\xi_2,v} \subseteq [\tau]_{\xi_1,\xi_2,v}\).

- Suppose \(\tau = \tau_1 \rightarrow \tau_2\). Let \(t \in [\tau_1]_{\xi_1,v} \cap [\tau_2]_{\xi_2,v}\). Let \(w \in [\tau_1]_{\xi_1,v}^{[\tau_2]_{\xi_2,v}}\). We have \(w \in [\tau_1]_{\xi_1,v} \cap [\tau_2]_{\xi_2,v}\). Hence there are \(w_1, w_2\) with \(tw \rightarrow^\infty w_1 \in [\tau_2]_{\xi_1,v}\) and \(tw \rightarrow^\infty w_2 \in [\tau_2]_{\xi_2,v}\). By confluence modulo \(\mathcal{U}\) there are \(w'_1, w'_2\) such that \(w'_1 \sim_\mathcal{U} w'_2\) and \(w_1 \rightarrow^\infty w'_1\). By Lemma 4.10 both \([\tau_2]_{\xi_1,v}^{[\tau_2]_{\xi_2,v}}\) are stable, and thus so is \([\tau_2]_{\xi_1,v} \cap [\tau_2]_{\xi_2,v}\).
Hence $w'_1 \in [\tau_2]_{\xi_1,\nu} \cap [\tau_2]_{\xi_2,\nu}$. By the inductive hypothesis $w'_1 \in [\tau_2]_{\xi_1 \cap \xi_2,\nu}$. This shows $[\tau]_{\xi_1,\nu} \cap [\tau]_{\xi_2,\nu} \subseteq [\tau]_{\xi_1 \cap \xi_2,\nu}$. \hfill \square

**Lemma A.2.** If $\xi_1, \xi_2$ are stable then $[\mu]_{\xi_1,\nu}^{\xi_1} \cap [\mu]_{\xi_2,\nu}^{\xi_2} \subseteq [\mu]_{\xi_1 \cap \xi_2,\nu}^{\xi_1}$.

**Proof.** Induction on $\kappa_1$. We may assume $\kappa_1 < \kappa_2$. If $\kappa_1 = 0$ then $[\mu]_{\xi_1,\nu}^{\xi_1} \cap [\mu]_{\xi_2,\nu}^{\xi_2} = \emptyset \cap [\mu]_{\xi_2,\nu}^{\xi_2} = [\mu]_{\xi_2,\nu}^{\xi_2}$.

If $\kappa_1$ is a limit ordinal then there exists $\kappa_0 < \kappa_1$ with $t \in [\mu]_{\xi_0,\nu}^{\xi_0}$ and we may use the inductive hypothesis.

If $\kappa_1 = \kappa_1 + 1$ then we may assume $\kappa_2 = \kappa_2 + 1$. Let $t \in [\mu]_{\xi_1,\nu}^{\xi_1} \cap [\mu]_{\xi_2,\nu}^{\xi_2}$. Then $t = ct_1 \ldots t_k$ with $t_i \in [\sigma_i]_{\xi_1,\nu} \cap [\sigma_i]_{\xi_2,\nu}$. Let $c \in \text{Constr}(\mu)$, $\text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_k)$ and (using Lemma 4.6)

$$
\zeta_i = \xi_i([\mu]_{\xi_{i-1},\nu}^{\xi_i}/A, [\alpha_1]_{\xi_{i-1},\nu}/B_1, \ldots, [\alpha_k]_{\xi_{i-1},\nu}/B_k)
$$

for $l = 1, 2$, and $\mu = d_\mu(\vec{\alpha})$, and $B_1, \ldots, B_k$ are the parameter type variables of $d_\mu$, and $A$ is the recursive type variable of $d_\mu$. By Lemma 4.10 the valuations $\xi_1, \xi_2$ are stable. By Lemma A.1 we have $t_i \in [\sigma_i]_{\xi_1 \cap \xi_2,\nu}$. Using the inductive hypothesis, Lemma A.1 and Lemma 4.8 we conclude that $t_i \in [\sigma_i]_{\xi_i,\nu}$ where

$$
\zeta = \xi([\mu]_{\xi_{i-1},\nu}^{\xi_i}/A, [\alpha_1]_{\xi_{i-1},\nu}/B_1, \ldots, [\alpha_k]_{\xi_{i-1},\nu}/B_k).
$$

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

$$
\zeta' = \xi_{\xi_1}[\mu]_{\xi_{i-1},\nu}^{\xi_1}/A, [\alpha_1]_{\xi_{i-1},\nu}/B_1, \ldots, [\alpha_k]_{\xi_{i-1},\nu}/B_k).
$$

Hence $t_i \in [\sigma_i]_{\zeta', \nu}$ by Lemma 4.8. But this by definition implies $t \in [\mu]_{\xi_{i-1},\nu}^{\xi_i}$. \hfill \square

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

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

**Proof.** We proceed by induction on $\tau$. So let $Z = \{\xi_n\}_{n \in \mathbb{N}}$ be stable and let $\Xi = \{\xi_n\}_{n \in \mathbb{N}}$ be $\nu$-hereditary with $\nu$ and semi-complete with $Z, t$, and let $\{t_n\}_{n \in \mathbb{N}}$ be a $\tau, Z$-sequence. By the definition of $t_\infty$ there are the following possibilities.

- If $\tau$ is closed then $t_\infty = t_0 \in [\tau]_{\tau,\nu} = [\tau]_{\tau,\nu}$, by Lemma 4.6.
- If $\tau = A$ then $t_\infty \in i(A) = \{\tau\}_{\tau,\nu}$ because $\Xi$ is semi-complete with $Z, t$.
- If $\tau = \mu^\infty$ with $\mu = d_\mu(\vec{\alpha})$ then let $\Xi' = \Xi[\tau]$ where $\tau = \{\tau_A\}_{A \in \mathcal{V}_T}$ with $\tau_A = \tau, \tau_{B_j} = \alpha_j$ and $\tau_{A'} = A'$ for $A' \notin \{A, B_1, \ldots, B_l\}$, where $B_1, \ldots, B_l$ are the parameter type variables of $d_\mu$, and $A$ is the recursive type variable of $d_\mu$. Note that $\Xi'$ is $\nu$-hereditary, because $\Xi$ is. Let $Z' = \{\xi_n\}_{n \in \mathbb{N}}$ where $\xi_n = \xi_{\tau}([\alpha_1]_{\zeta_n,\nu}/B_1, \ldots, [\alpha_k]_{\zeta_n,\nu}/B_k)$. Note that $Z' \subseteq \Xi'$ follows from Lemma 4.8, because $\zeta_n \subseteq \xi_n$ and thus $[\alpha_j]_{\xi_n,\nu} \subseteq [\alpha_j]_{\zeta_n,\nu}$. Also, $Z'$ is stable by Lemma 4.10, because $Z$ is. Let $t'(A') = \tau_{A'}$ for any $A'$. We show the following.

- Let $X = \{\xi_n\}_{n \in \mathbb{N}}$ be such that $\xi_n(A') = \xi_n(A')$ for $A' \neq A$. If $\Xi'$ is semi-complete with $X, t'$ then $\Xi'$ is semi-complete with $X', t'$ where $X' = \{\xi_n\}_{n \in \mathbb{N}}$ and $\lambda_n = \lambda_n([\Phi_{d_\nu, \zeta_n,\tau}(\lambda_n(A'))/A]$. 


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

$$
\chi'_n(A) = \Phi_{d_{\mu}, \zeta'_n, v}(\chi_n(A))
\subseteq \Phi_{d_{\mu}, \zeta'_n, v}(\lbrack d_{\mu} \rbrack_{\xi_n,v}^{\infty})
= \lbrack d_{\mu} \rbrack_{\xi_n,v}^{\infty}
= \zeta'_n(A)
$$

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 $w_{\infty} = f''(A', \Xi', \{w_n\}_{n \in \mathbb{N}}) \in \iota'(A')$. 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 $\Xi'$ is semi-complete with $X, \iota'$. 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 $w_n = cu_1^n \ldots u_n^k$ and $u_i^n \to \infty$ $w_{i+1}$, and $w_n \in \lbrack \sigma_i \rbrack_{\zeta_n[X(0)/A], v}$ where $\text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_k)$. Because $\chi_n(A') = \zeta'_n(A')$ for $A' \neq A$, we have $\zeta_n[X(A)/A] = \chi_n$. Hence $w_n \in \lbrack \sigma_i \rbrack_{\chi_n,v}$. Thus $\{u_i^n\}_{n \in \mathbb{N}}$ is a $\sigma_i, X'$-sequence. Note that $\sigma_i$ is smaller than $\tau$. Because $\Xi'$ is semi-complete with $X, \iota'$, by the inductive hypothesis we have $w_{\infty} = f''(\sigma_i, \Xi', \{u_i^n\}_{n \in \mathbb{N}}) \in \lbrack \sigma_i \rbrack_{\iota', v}$. Note that

$$
w_{\infty} = f''(A, \Xi', \{w_n\}_{n \in \mathbb{N}})
= f''(\tau A, \Xi, \{w_n\}_{n \in \mathbb{N}})
= f''(\tau, \Xi, \{w_n\}_{n \in \mathbb{N}})
= cu_1^n \ldots w_{\infty}.
$$

Hence $w_{\infty} \in \Phi_{d_{\mu}, \iota', v}(\iota'(A))$. We have $\iota'(A) = \lbrack \tau \rbrack_{\iota, v} = [d_{\mu}]_{\iota, v}^{\infty}$ by Lemma 4.6 because $\iota'(B_j) = \lbrack \alpha_j \rbrack_{\iota, v}$ for $j = 1, \ldots, l$. Hence $w_{\infty} \in \Phi_{d_{\mu}, \iota', v}(\lbrack d_{\mu} \rbrack_{\iota, v}^{\infty}) = [d_{\mu}]_{\iota, v}^{\infty} = [\tau]_{\iota, v} = \iota'(A)$. We have thus shown $(\ast)$. Let $Z^\kappa = \{\zeta_n^\kappa\}_{n \in \mathbb{N}}$ be such that $\zeta_n^\kappa = \zeta'_n[\lbrack \mu \rbrack_{\xi_n,v}/A]$. Then $Z^\kappa \subseteq \Xi'$ follows from Lemma 4.8. Also $Z^\kappa$ is stable by Lemma 4.10, because $Z, Z'$ are. By induction on $\kappa$ we show that $\Xi'$ is semi-complete with $Z^\kappa, \iota'$. We distinguish three cases.

- $\kappa = 0$. Then $\zeta_n^\kappa = \zeta'_n[\lbrack \mu \rbrack_{\xi_n,v}/A] = \zeta_n^0[0]/A$. We show that for every $A'$ and every $A', Z^0$-sequence $\{w_n\}_{n \in \mathbb{N}}$ we have $w_{\infty} = f''(\tau A', \Xi, \{w_n\}_{n \in \mathbb{N}}) \in \iota'(A') = \lbrack \tau A' \rbrack_{\iota, v}$. Let $\{w_n\}_{n \in \mathbb{N}}$ be a $A', Z^0$-sequence. If $A' \neq A$ then $\zeta_n^0(A') = \zeta_n^0(A) = \lbrack \tau A \rbrack_{\xi_n,v}$, 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 $\tau$.

Thus by the (main) inductive hypothesis we have $f''(\tau A', \Xi, \{w_n\}_{n \in \mathbb{N}}) \in \lbrack \tau A' \rbrack_{\iota, v}$, i.e., $w_{\infty} \in \iota'(A')$. If $A' = A$ then $\zeta_n^0(A) = 0$, so there does not exist a $A, Z^0$-sequence.

- $\kappa = \kappa' + 1$. Then

$$
\zeta_n^\kappa = \zeta_n^{\kappa'}[\lbrack \mu \rbrack_{\xi_n,v}^{\kappa'+1}/A]
= \zeta_n^{\kappa'}(\Phi_{d_{\mu}, \zeta'_n, v}(\lbrack \mu \rbrack_{\xi_n,v}^{\kappa'}))/A
= \zeta_n^{\kappa'}(\Phi_{d_{\mu}, \zeta'_n, v}(\zeta'_n(\alpha_j)))/A.
$$

By the inductive hypothesis $\Xi'$ is semi-complete with $Z^\kappa, \iota'$. Hence $\Xi'$ is semi-complete with $Z^\kappa, \iota'$ by $(\ast)$.

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

$$
w_{\infty} = f''(A', \Xi, \{w_n\}_{n \in \mathbb{N}})
= f''(\tau A', \Xi, \{w_n\}_{n \in \mathbb{N}}) \in \iota'(A').
$$
If $A' \neq A$ then the argument is the same as the one used in showing that $\\Sigma'$ is semi-complete with $Z^0, \iota'$. So assume $A' = A$. Then $w_n \in \zeta_n^0(A) = [\mu]_{\xi,n,v}^0$ for $n \in \mathbb{N}$. Since $\xi$ is a limit ordinal, for each $n \in \mathbb{N}$ there is $\zeta_n < \xi$ such that $w_n \in [\mu]_{\xi,n,v}^\zeta$. Because $w_0 \to^\omega w_n$ for $n \in \mathbb{N}$ and $\zeta_0, \zeta_n$ are stable, by Lemma A.2 we obtain $w_n \in [\mu]_{\xi,n,v}^\zeta$ for $n \in \mathbb{N}$. So $\{w_n\}_{n \in \mathbb{N}}$ is a $A, Z^0$-sequence. By the inductive hypothesis $\\Sigma'$ is semi-complete with $Z^{\omega_0}, \iota'$, so $w_\infty \in \iota'(A)$.

Now taking $\xi = \omega$ we conclude that $\\Sigma'$ is semi-complete with $Z^{\omega}, \iota'$. Note that $\zeta_n^\omega(A) = [\tau]_{\xi,n,v}^\omega$ for $n \in \mathbb{N}$. Hence $\{t_n\}_{n \in \mathbb{N}}$ is a $A, Z^{\omega}$-sequence, because it is a $\tau, Z$-sequence and $\tau_A = \tau$. Therefore $t_\infty = f^\nu(\tau, \xi, \{t_n\}_{n \in \mathbb{N}}) \in \iota'(A) = [\tau]_{\xi,n,v}^\omega$.

- If $\tau = \nu_0^\infty$ with $\nu_0 = d_{\nu_0}(\bar{\alpha})$ then let $\\Sigma' = \Xi[\tau]$ where $\tau = \{\tau_A\}_{A \in V_T}$ with $\tau_A = \tau, \kappa, \nu_0$ such that $\tau = \kappa, \nu_0, \alpha_j$ and $\Phi_A = A'$ for $A' \notin \{A, B_1, \ldots, B_l\}$, where $B_1, \ldots, B_l$ are the parameter type variables of $d_{\nu_0}$, and $A$ is the recursive type variable of $d_{\nu_0}$. Note that $\\Sigma'$ is $\nu$-hereditary, because $\Xi$ is.

Let $Z' = \{\zeta_n'\}_{n \in \mathbb{N}}$ where $\zeta_n'(A') = [\tau_A'_{\xi,n,v}^\omega]$ for all $A'$. Note that $Z' \subseteq \Xi'$ follows from Section 4.8, because $\zeta_n \subseteq \zeta_n$ and thus $[\tau_A'_{\xi,n,v}^\omega] \subseteq [\tau_A'_{\xi,n,v}^\omega]$. Also $Z'$ is stable by Lemma 4.10, because $Z$ is. Let $\iota'(A) = [\tau_A'_{\xi,n,v}^\omega]$ for all $A'$. We show the following.

($\ast$) Let $t_0$ be a type variable valuation such that $t_0(A) = \iota'(A')$ for $A' \neq A$. If $\\Sigma'$ is semi-complete with $Z', t_0$ then $\\Sigma'$ is semi-complete with $Z', t_1$ where $t_1 = t_0[\Phi_{d_{\nu_0},v_0,v}(t_0(A))/A]$. 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 $w_\infty = f^{\nu}(\tau, \Xi', \{w_n\}_{n \in \mathbb{N}}) \in \iota'(A')$. So let $\{w_n\}_{n \in \mathbb{N}}$ be a $A', Z'$-sequence, i.e. $w_n \in \{A\}_{\xi,n,v} = \zeta_n'(A')$ and $w_\infty \to^\omega w_{n+1}$. If $A' \neq A$ then $\iota'(A) = t_0(A')$. Because $\\Sigma'$ is semi-complete with $Z', t_0$, we have $w_\infty \in \iota_0(A') = \iota_1(A')$. If $A' = A$ then $w_n \in \{\tau\}_{\xi,n,v} = \Phi_{d_{\nu_0},v_0,v}([\tau]_{\xi,n,v}^\omega)$. Hence $w_n = c w_1 \ldots w_k$ and $w_n \in \{\sigma\}_{\xi,n,v}^\omega$ and $w_\infty \to^\omega w_{n+1}$. Then $c \in \text{Constr}(\nu_0)$, ArgTypes$(c) = (\sigma_1, \ldots, \sigma_k)$. Thus $\{w_n\}_{n \in \mathbb{N}}$ is a $\sigma_i, Z'$-sequence. Because $\\Sigma'$ is semi-complete with $Z', t_0$ and $\sigma_i$ is smaller than $\tau$, by the inductive hypothesis $w_\infty = f^{\nu}(\sigma_i, \Xi', \{w_n\}_{n \in \mathbb{N}}) \in \{\sigma_i\}_{\xi,n,v}^\omega$. Note that $w_\infty = c w_1 \ldots w_k$ by Definition 5.6. Hence $w_\infty \in \Phi_{d_{\nu_0},v_0,v}(t_0(A)) = \iota_1(A)$ by Corollary 4.7. We have thus shown ($\ast$).

Let $t_0 = \iota'[T^{\infty}/A]$. We show that $\\Sigma'$ is semi-complete with $Z', t_0$. 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 $w_n \in \{A\}_{\xi,n,v} = \{\tau_A\}_{\xi,n,v}$. If $A' \neq A$ then $w_n \in \zeta_n'(A') = [\tau_A'_{\xi,n,v}^\omega]$ and so $\{w_n\}_{n \in \mathbb{N}}$ is a $\tau_A', Z$-sequence. Since $\tau_A'$ is smaller than $\tau$ (because $\tau_A' = A'$ or $\tau_A' = \alpha_j$) and $\Xi'$ is semi-complete with $Z, t_0$ by the inductive hypothesis $w_\infty = f^{\nu}(A', \Xi', \{w_n\}_{n \in \mathbb{N}}) = f^{\nu}(\tau_A', \Xi, \{w_n\}_{n \in \mathbb{N}}) \in [\tau_A'_{\xi,n,v}^\omega]$. If $A' = A$ then $\iota_0(A) = T^{\infty}$, so $w_\infty \in \iota_0(A)$.

Now let $\tau_{\zeta} = \iota'[\nu_0_{I,v}^{\infty}/A]$ for an ordinal $\zeta$ (recall that $[\nu_0_{I,v}^{\infty} = T^{\infty}$). We show by induction on $\zeta$ that $\\Sigma'$ is semi-complete with $Z', \tau_{\zeta}$. For $\zeta = 0$ we have shown this in the previous paragraph. If $\zeta = \zeta + 1$ then this follows from ($\ast$) because $\Phi_{d_{\nu_0},v_0,v}(\iota_{\zeta}(A)) = \Phi_{d_{\nu_0},v_0,v}(\iota_{\zeta}(A)) = \Phi_{d_{\nu_0},v_0,v}(\iota_{\zeta}(A)) = \nu_0_{I,v}$ by Corollary 4.7 and Lemma 4.6 (note that $\tau_{\zeta}(B_j) = \iota_{\zeta}(B_j) = [\alpha_j]_{I,v}$). So let $\zeta$ 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 $\\Sigma'$ is semi-complete with $Z', \tau_{\zeta}$ for $\zeta < \zeta$. So if $A' = A$ then $w_\infty \in \bigcap_{\zeta < \zeta'} \iota_{\zeta}(A)$ for $\zeta' < \zeta$. So if $A' = A$ then $w_\infty \in \iota_0(A) = \iota_{\zeta}(A)$.

Now because $\{t_n\}_{n \in \mathbb{N}}$ is a $\tau, Z$-sequence and $\tau_A = \tau$, the sequence $\{t_n\}_{n \in \mathbb{N}}$ is also a $A, Z'$-sequence. Because $\\Sigma'$ is semi-complete with $Z', \iota_\infty$ 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 $t_\infty \in t_\infty(A) = [t_0]_v = [T]_{t,v}$.

- If $\tau = \forall i. \tau'$ with $i$ fresh then $t_\infty = t_0$. We need to show $t_0 \in [T]_{t,v}$. Let $\varkappa \in \Omega$. For $n \in \mathbb{N}$ we have $t_0 \to ^\infty t_n \in [T]_{z_0,v}$, so for each $n \in \mathbb{N}$ there exists $t'_n$ with $t_0 \to ^\infty t'_n \in [T]_{z_0,v[\varkappa/i]}$. Because $\mathcal{Z}$ is stable, by Lemma 5.10 there is a sequence $\{t''_n\}_{n \in \mathbb{N}}$ such that $t_0 \to ^\infty t''_0$ and $t''_n \to ^\infty t''_{n+1}$ and $t''_n \in [T]_{z_0,v[\varkappa/i]}$ for $n \in \mathbb{N}$. Hence $\{t''_n\}_{n \in \mathbb{N}}$ is a $\tau$, $\mathcal{Z}$-sequence (with $v[\varkappa/i]$). The family $\mathcal{X}$ is $\nu$-hereditary with $v[\varkappa/i]$ by Lemma 5.3. Because $\mathcal{X}$ is also semi-complete with $\mathcal{Z}, i$, by Remark 5.12 and the inductive hypothesis there is $t'\infty$ with $t_0 \to ^\infty t'\infty \in [T]_{t,v'\varkappa}$. Because $\varkappa \in \Omega$ was arbitrary, this implies $t_0 \in [T]_{t,v}$.

- If $\tau = \tau_1 \rightarrow \tau_2$ with $\tau_1$ closed and $\tau_2$ strictly positive, then $t_\infty = t_0$. We need to show $t_0 \in [T]_{t,v}$. For $n \in \mathbb{N}$ we have $t_0 \to ^\infty t_n \in [T]_{z_0,v}$. Let $r \in [\tau_1]_{t,v}$. Because $\tau_1$ is closed, by Lemma 4.6 we have $r \in [\tau_1]_{z_0,v}$ for each $n \in \mathbb{N}$. Hence for each $n \in \mathbb{N}$ there is $t'_n$ with $t_0 \to ^\infty t_n \to ^\infty t'_n \in [\tau_2]_{z_0,v}$. Because $\mathcal{Z}$ is stable, by Lemma 5.10 there is a sequence $\{t''_n\}_{n \in \mathbb{N}}$ such that $t_0 \to ^\infty t''_0$ and $t''_n \to ^\infty t''_{n+1}$ and $t''_n \in [\tau_2]_{z_0,v}$ for $n \in \mathbb{N}$. Hence $\{t''_n\}_{n \in \mathbb{N}}$ is a $\tau_2$, $\mathcal{Z}$-sequence. By the inductive hypothesis there is $t'\infty \in [\tau_2]_{t,v}$ with $t''_0 \to ^\infty t'\infty$. Since $t_0 \to ^\infty t''_0$, also $t_0 \to ^\infty t'_\infty$. We have thus shown $t_0 \in [T]_{t,v}$. \qed
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.** $[\tau[r'/A]]_{\xi,v} = [\tau][\xi][r'/A]_{\xi,v}$.  

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

If $\tau = A$ then $[\tau[r'/A]]_{\xi,v} = [\tau][\xi,v]$. Let $n \in \Omega$. There is $t'$ such that $t \vdash t' \in [\tau[r'/A]]_{\xi,v}$. By the inductive hypothesis $t' \in [\tau][\xi][r'/A]_{\xi,v}$. This implies $t \in [\tau][\xi][r'/A]_{\xi,v}$, using Lemma 4.5 (we may assume $i \not\in FSV(\tau)$). The other direction is analogous.

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

**Lemma B.2.** $[\tau[s/i]]_{v} = [\tau][v(s)/i]$.  

*Proof.* Induction on $\tau$.  

**Theorem 7.3** (Soundness). If $\Gamma \vdash t : \tau$ with $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$ then for every size variable valuation $v : V_S \rightarrow \Omega$ and all $t \in [\tau_1]_v, \ldots, t_n \in [\tau_n]_v$ there exists $t'$ such that $t[t_1/x_1, \ldots, t_n/x_n] \rightarrow t' \in [\tau]_v$.

*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 $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau'$ because of $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau$ and $\tau \subseteq \tau'$. Let $t_1 \in [\tau_1]_v, \ldots, t_n \in [\tau_n]_v$. By the inductive hypothesis there is $t'$ with $t[t_1/x_1, \ldots, t_n/x_n] \rightarrow t' \in [\tau]_v$. By Lemma 7.2 we also have $t' \in [\tau']_v$.

(con) Assume $\Gamma \vdash cr_1 \ldots r_n : \rho^{s+1}$ because of $\Gamma \vdash r_k : \sigma_k[\rho^s/A][\alpha/B]$ for $k = 1, \ldots, n$ and $\text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_n)$ and $\text{Def}(c) = d$ and $\rho = d(\alpha)$ and $\Gamma = x_1 : \tau_1, \ldots, x_m : \tau_m$. Let $t_1 \in [\tau_1]_v, \ldots, t_m \in [\tau_m]_v$. By the inductive hypothesis for $k = 1, \ldots, n$ there is $t'_k$ with $r_k[t_1/x_1, \ldots, t_n/x_n] \rightarrow t'_k \in [\sigma_k[\rho^s/A][\alpha/B]]_v$. By Lemma B.1 and Lemma 4.6 we have $t'_k \in [\sigma_k[\rho^s[A][\alpha]/\alpha]]_v$ where $\xi(B_j) = [\alpha_j]_v$. Hence

| $cr_1 \ldots r_n|t_1/x_1, \ldots, t_m/x_m| \rightarrow cr'_1 \ldots r'_n \in [\rho]_v^{s+1}$.

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

| $\lambda x : \alpha.t|t_1/x_1, \ldots, t_n/x_n|r \rightarrow t' \in [\beta]_v$.

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


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

$$|t| |t_1/x_1, \ldots, t_n/x_n| \rightarrow_{\infty} r \in [\alpha \rightarrow \beta]_v$$

and

$$|t'| |t_1/x_1, \ldots, t_n/x_n| \rightarrow_{\infty} r' \in [\beta]_v.$$  

Hence there is $r''$ with $|tt'| |t_1/x_1, \ldots, t_n/x_n| \rightarrow_{\infty} rr' \rightarrow_{\infty} r'' \in [\beta]_v$.

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

(gen) Assume $\Gamma \vdash \Delta tv : \forall i. \tau$ because of $\Gamma \vdash t : \tau$ with $i \notin \text{FSV}(\Gamma)$ and $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$. Let $t_1 \in [\tau_1]_v, \ldots, t_n \in [\tau_n]_v$. Let $\kappa \in \Omega$. Since $i \notin \text{FSV}(\Gamma)$ by Lemma 4.5 we have $t_k \in [\tau_k]_{v[\kappa/i]}$ for $k = 1, \ldots, n$. By the inductive hypothesis there is $r_{\kappa
u}$ with $|\Delta tv| |t_1/x_1, \ldots, t_n/x_n| = |t| |t_1/x_1, \ldots, t_n/x_n| \rightarrow_{\infty} r_{\kappa \nu} \in [\tau]_{v[\kappa/i]}$. This implies

$$|t| |t_1/x_1, \ldots, t_n/x_n| \in [\forall i. \tau]_v.$$ 

(case) Assume $\Gamma \vdash \text{case}(t; \{c_k x_k^k \Rightarrow t_k | k = 1, \ldots, n\}) : \tau$ because of $\Gamma \vdash t : \rho^{s+1}$ and $\Gamma, x_k : \delta_k^k, \ldots, x_k^k : \delta_k^k \vdash t_k : \tau$ and $\text{ArgTypes}(c_k) = (\sigma_k^k, \ldots, \sigma_k^{nk})$ and $\delta_k^k = \sigma_k^k [\rho^s / \alpha] [\beta / \beta]$ and $\rho = d(\alpha)$ and $\Gamma = x_1 : \tau_1, \ldots, x_m : \tau_m$. Let $r_1 \in [\tau_1]_v, \ldots, r_m \in [\tau_m]_v$. By the inductive hypothesis there is $u$ with $|t| |r_1/x_1, \ldots, r_m/x_m| \rightarrow_{\infty} u \in [\rho^{s+1}]_v = [\rho^s]_v = [\rho^s]_v^{s+1}$ (note that we may have $v(s+1) = v(s) = \infty$, but then the last equation still holds because $\infty$ is the fixpoint ordinal). Hence $u = c_k u_i \ldots u_m$ where $u_i \in [\sigma_k^k]_{\xi, v}$ and $\xi(B_j) = [\alpha_j]_v$ and $\xi(A) = [\rho^s]_v^{s+1}$. Then by Lemma B.1 we have $u_i \in [\sigma_k^k]_{\xi, v} = [\delta_k^k]_v$. Hence by the inductive hypothesis there is $w$ with $|t_k |r_1/x_1, \ldots, r_m/x_m, u_i/x_k^1, \ldots, u_m/x_k^m| \rightarrow_{\infty} w \in [\tau]_v$. Note that also (we may assume $x_k^1, \ldots, x_k^m \notin \text{TV}(r_1, \ldots, r_m)$):

$$\text{case}(t; \{c_k x_k^k \Rightarrow t_k | k = 1, \ldots, n\}) |r_1/x_1, \ldots, r_m/x_m|$$

$$\rightarrow_{\infty} \text{case}(u; \{c_k x_k^k \Rightarrow t_k |r_1/x_1, \ldots, r_m/x_m|\})$$

$$\rightarrow |t_k |r_1/x_1, \ldots, r_m/x_m, u_i/x_k^1, \ldots, u_m/x_k^m|$$

$$\rightarrow_{\infty} w.$$ 

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

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

where $i \notin \text{FSV}(\Gamma, \mu, \tau, j_1, \ldots, j_m)$ and $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$. Let $t_1 \in [\tau_1]_v, \ldots, t_n \in [\tau_n]_v$. Let $t' = |t| |t_1/x_1, \ldots, t_n/x_n|$ and $r = Y(\lambda f. t')$. Note that $r = |\text{fix } f : \forall j_1 \ldots j_m. \mu \rightarrow \tau. t| |t_1/x_1, \ldots, t_n/x_n|$. By induction on $\kappa \in \Omega$ we show $r \in [\forall j_1 \ldots j_m. \mu \rightarrow \tau]_{v[\kappa/i]}$. Let $\kappa_1, \ldots, \kappa_m \in \Omega$. We need to show that for every $u \in [\mu]_{v[\kappa]}$ there is $r'$ with $ru \rightarrow_{\infty} r' \in [\tau]_{v'[\kappa]}$, where $v' = v[\kappa_1/j_1, \ldots, \kappa_m/j_m]$ (recall $i \notin \text{FSV}(\mu, \tau)$). There are three cases.

- $\kappa = 0$. Then $[\mu]_{v[\kappa]} = \emptyset$.
- $\kappa = \kappa + 1$. By the inductive hypothesis for $\kappa$ we have $r \in [\forall j_1 \ldots j_m. \mu^i \rightarrow \tau]_{v[\kappa'/i]}$. By the main inductive hypothesis $t'[r'/f] \in [\forall j_1 \ldots j_m. \mu^i \rightarrow \tau]_{v[\kappa'/i]}$. Let $u \in [\mu]_{v[\kappa']}$. Then there is $r'$ with $ru \rightarrow^{*} t'[r/f]u \rightarrow_{\infty} r' \in [\tau]_{v'}$. 
We have thus shown that \( r \in [\forall j_1 \ldots j_m. \mu^i \to \tau]_{v[\kappa/i]} \) for all \( \kappa \in \Omega \). In particular, this holds for \( \kappa = \infty \), which implies \( r \in [\forall j_1 \ldots j_m. \mu \to \tau]_v \), 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 \( \Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n \). Let \( t_1 \in [\tau_1]_v, \ldots, t_n \in [\tau_n]_v \). Let \( t' = |t| t_1/x_1, \ldots, t_n/x_n \) and \( r = Y(\lambda f.t') \). 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 \( r_n \rightarrow \tau' n' \in [\tau']_{v[n/j]} \).

For \( n = 0 \), we have \( r_0 = r \in [\tau]_{v[0/j]} \) directly from definitions and the fact that \( j \notin \text{SV}(\tau) \), because \( [\nu^{\min(s,j)}]_{v'[0/j]} = \left[\nu\right]_{v'[0/j]} = \top^\infty \) for any \( v' \). So assume \( r_n \rightarrow \tau'' n'' \in [\tau'']_{v[n/j]} \). Because \( \Gamma, f : \tau' \vdash t : \tau'' \), by the main inductive hypothesis there is \( t'' \) with \( t'[r_n'/f] \rightarrow \tau'' n'' \in [\tau'']_{v[n/j]} \). We have \( r_{n+1} = t'[r_n/f] \rightarrow \tau'' n'' \rightarrow t'' \). Take \( r_{n+1}' = t'' \).

Because \( r_{n+1}' \in [\tau']_{v[n/j]} \) and \( j \notin \text{SV}(\tau) \), it follows from definitions and Lemma B.2 that \( r_{n+1}' \in [\tau']_{v[n+1/j]} \).

We have thus shown that for each \( n \in \mathbb{N} \) there exists \( r_n' \) such that

\[ r_n \rightarrow \tau' n' \in [\text{chgtgt}(\tau, \nu^{\min(s,j)})]_{v[n/j]} \]

Now by Lemma 7.1 there is \( r' \) with

\[ |\text{cofix} j f : \tau.t|[t_1/x_1, \ldots, t_n/x_n] = r \rightarrow \tau' n' \in [\tau]_v. \]
APPENDIX C. TYPE CHECKING AND TYPE INFERENCE

In this section we show that type checking in $\lambda^\circ$ 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 $\Gamma$, then there exists a minimal type $T(\Gamma; t)$ such that $\Gamma \vdash t : T(\Gamma; t)$ and for every type $\tau$ with $\Gamma \vdash t : \tau$ we have $T(\Gamma; t) \sqsubseteq \tau$. To define $T(\Gamma; t)$ 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.

\begin{itemize}
  \item $(\alpha \to \beta) \sqcup (\alpha' \to \beta') = (\alpha \sqcap \alpha') \to (\beta \sqcup \beta')$.
  \item $(\alpha \to \beta) \sqcap (\alpha' \to \beta') = (\alpha \sqcap \alpha') \to (\beta \sqcap \beta')$.
  \item $(\forall i.\alpha) \sqcup (\forall i.\alpha') = \forall i.\alpha \sqcup \alpha'$.
  \item $(\forall i.\alpha) \sqcap (\forall i.\alpha') = \forall i.\alpha \sqcap \alpha'$.
  \item $(d^\alpha_\mu(\bar{\alpha})) \sqcup (d^\alpha_\mu(\bar{\beta})) = d^\max(s,s')(\bar{\gamma})$ where $\gamma_i = \alpha_i \sqcup \beta_i$.
  \item $(d^\alpha_\mu(\bar{\alpha})) \sqcap (d^\alpha_\mu(\bar{\beta})) = d^\min(s,s')(\bar{\gamma})$ where $\gamma_i = \alpha_i \sqcap \beta_i$.
  \item $(d^\alpha_\nu(\bar{\alpha})) \sqcap (d^\alpha_\nu(\bar{\beta})) = d^\min(s,s')(\bar{\gamma})$ where $\gamma_i = \alpha_i \sqcap \beta_i$.
  \item $(d^\alpha_\nu(\bar{\alpha})) \sqcup (d^\alpha_\nu(\bar{\beta})) = d^\max(s,s')(\bar{\gamma})$ where $\gamma_i = \alpha_i \sqcup \beta_i$.
\end{itemize}

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

Proof. By induction on $\tau_i$. \hfill \Box

Lemma C.3. If $\tau$ is strictly positive and $\tau \sqsubset \tau'$ 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 $\tau$. \hfill \Box

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. \hfill \Box

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(s1+1, s2+1) = max(s1, s2) + 1, max(∞, s) = ∞, 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 $\bar{s}$ 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 $\bar{s} + 1$ (note that $s'' \geq 1$).

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

Lemma C.6.

\begin{enumerate}
  \item If $s \geq 1$ then $s \geq \bar{s} + 1$.
\end{enumerate}
(2) \( s \leq s + 1 \).

**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) \( s_1 \leq s_2 \).

**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

\[
\begin{align*}
\max(\min(s_1, s_2), s_3) &= \min(\max(s_1, s_3), \max(s_2, s_3)) \\
\min(\max(s_1, s_2), s_3) &= \max(\min(s_1, s_3), \min(s_2, s_3))
\end{align*}
\]

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

\[
\begin{align*}
\min(s_1, s_2) \geq s_3 &\iff s_1 \geq s_3 \land s_2 \geq s_3 \\
\max(s_1, s_2) \leq s_3 &\iff s_1 \leq s_3 \land s_2 \leq s_3
\end{align*}
\]

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 \( \overline{s} \geq \overline{s_0} \) for \( s, s_0 \) of the form as above. Define \( s'_0 \) by replacing in \( s_0 \) each \( i \in SV(s) \setminus SV(\overline{s}) \) (i.e. each size variable which occurs only superfluously in \( s \)) with \( \infty \), and simplifying using obvious identities. First note that we may assume that some \( i \in SV(s) \setminus SV(\overline{s}) \) occurs in \( s_0 \), because otherwise \( \overline{s} \geq \overline{s_0} \) follows from \( s \geq s_0 + 1 \) by setting each \( i \in SV(s) \setminus SV(\overline{s}) \) 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 \( i \in SV(s) \setminus SV(\overline{s}) \). Hence, we may assume \( v(i) = v(\overline{s}) + 1 \) for \( i \in SV(s) \setminus SV(\overline{s}) \). Then \( v(s) = max(v(i), v(\overline{s} + 1)) = max(v(\overline{s}) + 1, v(\overline{s} + 1)) = v(\overline{s}) + 1 \) where \( i \in SV(s) \setminus SV(\overline{s}) \) (by how \( \overline{s} \) is obtained from \( s \)). Also \( v(s_0) \geq \min(v(i), v(s'_0)) \) where \( i \in SV(s) \setminus SV(\overline{s}) \). 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 \( \overline{s} \leq s_0 \).

**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 \( \min(i, s_1, s_2, \ldots) \leq s_0 + 1 \) then \( \min(i + 1, s_1, s_2, \ldots) \leq s_0 + 1 \) with \( s_0 \) of the form \( \max(\ldots) \) as above. We may assume \( i \notin SV(s_1, s_2, \ldots) \). There are two cases.

- \( i \notin SV(s_0) \). Suppose \( v(s_0) + 1 < \min(v(i) + 1, v(s_1), v(s_2), \ldots) \). Then \( v'(s_0) + 1 = v(s_0) + 1 < \min(v(i) + 1, v(s_1), v(s_2), \ldots) = v'(\min(i, s_1, s_2, \ldots)) \) for \( v' = v[i + 1/i] \). Contradiction.

- \( s_0 = \max(i + c, s'_1, s'_2, \ldots) \). Then \( v(s_0) + 1 = \max(v(i) + c, v(s'_1), v(s'_2), \ldots) + 1 \geq v(i) + c + 1 \geq v(i) + 1 \geq v(\min(i + 1, s_1, s_2, \ldots)) \). \( \square \)
To save on notation we introduce a dummy ⊥ type and set ⊥ ▐ τ = τ ▐ ⊥ = τ. The dummy type ⊥ 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 ∈ Constr(d) such that B ∈ TV(σi) for some i, where ArgTypes(c) = (σ1, . . . , σn). 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 τ(Γ; t) of t in Γ.

- τ(Γ; x: τ; x) = τ.
- τ(Γ; ct1 . . . tn) = µmax(s1,...,sn)+1 if ArgTypes(c) = (σ1, . . . , σn) and µ = dµ(τ1, . . . , τm) and Def(c) = dµ and τ(Γ; ti) = σi[dµ(α1,...,αm)/A][β1/β1,...,βm/βm] (we take si = 0 and α1 = ⊥ if A /∈ TV(σi), and β1 = ⊥ if B /∉ TV(σi)) and σi ⊂ σi and τj = ∩i=1(αj ∪ βj).

Note that τj ≠ ⊥ because of our assumption on the occurrences of Bj.

- τ(Γ; ct1 . . . tn) = νmin(s1,...,sn)+1 if ArgTypes(c) = (σ1, . . . , σn) and ν = dν(τ1, . . . , τm) and Def(c) = dν and τ(Γ; ti) = σi[dν(α1,...,αm)/A][β1/β1,...,βm/βm] (we take si = ∞ and α1 = ⊥ if A /∈ TV(σi), and β1 = ⊥ if B /∉ TV(σi)) and σi ⊂ σi and τj = ∩i=1(αj ∪ βj).

- τ(Γ; λx : α.t) = α → β if τ(Γ, x : α; t) = β.
- τ(Γ; tt′) = β if τ(Γ; t) = α → β and τ(Γ; t′) ⊂ α.
- τ(Γ; ts) = τ[s/i] if τ(Γ; t) = ∀i.τ.
- τ(Γ; Ai.t) = ∀i.τ if τ(Γ; t) = τ and i /∈ FSV(Γ).
- τ(Γ; case(t; {c_kx_k ⇒ t_k | k = 1, . . . , n})) = τ if τ(Γ; t) = µs and µ = d(β) and ArgTypes(c_k) = (σ1_k,...,σn_k) and δ_k = σ_k[µs/A][β/β] and τ(Γ, x_1 : δ_k,...,x_k : δ_k ; t_k) = τ_k and τ = ∩k=1n τ_k.

- τ(Γ; case(t; {c_kx_k ⇒ t_k | k = 1, . . . , n})) = τ if τ(Γ; t) = νs and s ≥ 1 and ν = d(β) and ArgTypes(c_k) = (σ1_k,...,σn_k) and δ_k = σ_k[νs/A][β/β] and τ(Γ, x_1 : δ_k,...,x_k : δ_k ; t_k) = τ_k and τ = ∩k=1n τ_k.

- τ(Γ; fix f : ∀j1 . . . jm.µ → τ.t) = ∀j1 . . . jm.µ → τ if

  τ(Γ, f : ∀j1 . . . jm.µ → τ.t) ⊂ ∀j1 . . . jm.µ+1 → τ

  and i /∈ FSV(Γ, µ, τ, j1, . . . , jn).

- τ(Γ; cofix f : τ.t) = τ if

  τ(Γ, f : chg(t, µmin(s,j))/τ.t) ⊂ chg(t, µmin(s,j+1))

and tgt(τ) = ν and j /∈ FSV(Γ) and j /∈ SV(τ).

In other cases not accounted for by the above points τ(Γ; t) is undefined. In particular, if the result of the operation ▐ is not defined then τ(Γ; t) is undefined. Note that if τ(Γ; t) is defined then it is uniquely determined.

**Lemma C.11.** If τ(Γ; t) is defined then Γ ⊩ t : τ(Γ; t).

**Proof.** Induction on the definition of τ(Γ; t), using Lemma C.2, Lemma C.3 and Lemma C.6. We show a few representative cases in detail.

- τ(Γ, x : τ; x) = τ. Then Γ, x : τ ⊩ x : τ.
- τ(Γ; ct1 . . . tn) = µmax(s1,...,sn)+1 where ArgTypes(c) = (σ1, . . . , σn) and µ = dµ(τ1, . . . , τm) and Def(c) = dµ and τ(Γ; ti) = σi[dµ(α1,...,αm)/A][β1/β1,...,βm/βm] and σi ⊂ σi and τj = ∩i=1(αj ∪ βj). We have Γ ⊩ ti : σi[dµ(α1,...,αm)/A][β1/β1,...,βm/βm] by
the inductive hypothesis. By Lemma C.2 we have $\alpha^i_j \subseteq \tau_j$. Hence, by Lemma C.3 and the (sub) typing rule, $\Gamma \vdash t_i : \sigma_i[d_{\mu}^{\alpha_1(s_1,...,s_n)}(\tau_1,...,\tau_m)/A][\tau_1/B_1,...,\tau_m/B_m]$. Thus $\Gamma \vdash t_1 \ldots t_n : \mu^{\alpha(s_1,...,s_n)+1}$ by the (con) typing rule.

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

Lemma C.12.

(1) For any type $\tau$ we have $\tau \subseteq \tau$.
(2) If $\tau_1 \sqcup \tau_2 \sqsubseteq \tau_3$ 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 $\tau_2 = d_{\mu}^s(\beta)$ then $\tau_1 = d_{\mu}^s(\alpha)$ and $\tau_3 = d_{\mu}^s(\gamma)$ and $s_1 \leq s_2 \leq s_3$ and $\alpha_k \subseteq \beta_k \subseteq \gamma_k$. By the inductive hypothesis $\alpha_k \subseteq \gamma_k$. 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 $\alpha_3 \subseteq \alpha_2 \subseteq \alpha_1$ and $\beta_1 \subseteq \beta_2 \subseteq \beta_3$. By the inductive hypothesis $\alpha_3 \subseteq \alpha_1$ and $\beta_1 \subseteq \beta_3$. Thus $\tau_1 \sqsubseteq \tau_3$.

Lemma C.13.

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

Proof. We show both points simultaneously by induction on $\tau$.

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

Assume $\tau_1 = \forall i. \alpha_1$, $\tau_2 = \forall i. \alpha_2$ and $\tau = \forall i. \gamma$. Then $\tau_1 \cup \tau_2 = \forall i. \alpha_1 \sqcup \alpha_2$. Because $\alpha_1 \sqcup \alpha_2 \sqsubseteq \gamma$, by the inductive hypothesis $\alpha_1 \sqsubseteq \gamma$ and $\alpha_2 \sqsubseteq \gamma$. Thus $\tau_1 \sqsubseteq \tau$ and $\tau_2 \sqsubseteq \tau$.

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

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

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) $\tau_1 \cap \tau_2 \sqsubseteq \tau_1' \cap \tau_2'$.

Proof. We show both points simultaneously by induction on $\tau_1$.

(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 $\tau_1 \cup \tau_2 = (\alpha_1 \sqcup \alpha_2) \rightarrow (\beta_1 \sqcup \beta_2)$ and $\tau_1' \cup \tau_2' = (\alpha_1' \sqcup \alpha_2') \rightarrow (\beta_1' \sqcup \beta_2')$. By the inductive hypothesis $\alpha_1' \sqcup \alpha_2' \sqsubseteq \alpha_1 \sqcup \alpha_2$ and $\beta_1' \sqcup \beta_2' \sqsubseteq \beta_1 \sqcup \beta_2$. Hence $\tau_1 \cup \tau_2 \sqsubseteq \tau_1' \cup \tau_2'$.

(2) The proof for the second point is analogous to the first one.
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 $\alpha \subseteq \alpha'$ and $\beta \subseteq \beta'$. By the inductive hypothesis $\alpha \cup \beta \subseteq \alpha' \cup \beta'$. Thus $\tau_1 \cup \tau_2 \subseteq \tau_1' \cup \tau_2'$.

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

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

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

**Corollary C.15.** If $\tau_1 \subseteq \tau$ and $\tau_2 \subseteq \tau$ then $\tau_1 \cup \tau_2 \subseteq \tau$.

**Proof.** One shows by induction that $\tau \subseteq \tau \cap \tau$ and $\tau \cup \tau \subseteq \tau$. Then the corollary follows from Lemma C.14 and Lemma C.12.

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

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

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

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

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

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

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

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

$$\mathcal{T}(\Gamma; t_i) = \sigma_i'[d^i_{\mu}(\alpha_1, \ldots, \alpha_m)/A][\beta_1/B_1, \ldots, \beta_m/B_m]$$

(we take $s_i = 0$ and $\alpha_i = {\bot}$ if $A \notin \text{TV}(\sigma_i)$, and $\beta_i = {\bot}$ if $B_j \notin \text{TV}(\sigma_i)$) and $\tau_i \subseteq \sigma_i$ and $\tau_j = \bigcup_{i=1}^{n}(\alpha_i \cup \beta_j)$. By the inductive hypothesis

$$\mathcal{T}(\Gamma'; t_i) \subseteq \mathcal{T}(\Gamma; t_i) = \sigma_i'[d^i_{\mu}(\alpha_1, \ldots, \alpha_m)/A][\beta_1/B_1, \ldots, \beta_m/B_m].$$

By Lemma C.3 we have $\mathcal{T}(\Gamma'; t_i) = \rho_i[d^i_{\mu}(\gamma_1, \ldots, \gamma_m)/A][\delta_1/B_1, \ldots, \delta_m/B_m]$ with $\gamma_i \subseteq \alpha_i$ and $\delta_i \subseteq \beta_j$ and $s_i \leq s_i$. Since $\rho_i \subseteq \sigma_i'$ by Lemma C.12 we obtain $\rho_i \subseteq \sigma_i'$. Let $\tau'_j = \bigcup_{i=1}^{n}(\gamma_i \cup \delta_j)$. Thus $\mathcal{T}(\Gamma'; ct_1 \ldots t_n) = \mu^{\max(s'_1+1,\ldots,s'_n+1)}$ where $\mu = d_{\mu}(\tau_1, \ldots, \tau_n)$. By Lemma C.14 we have $\tau'_j \subseteq \tau_j$. Also $\max(s'_1+1, \ldots, s'_n+1) \leq \max(s_1+1, \ldots, s_n+1)$. Hence $\mathcal{T}(\Gamma'; ct_1 \ldots t_n) \subseteq \mathcal{T}(\Gamma; ct_1 \ldots t_n)$.

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

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

11:42
Theorem C.17. \( \Gamma \vdash t : \tau \) iff \( \Gamma \vdash t : T(\Gamma; t) \) and \( T(\Gamma; t) \subseteq \tau \).

Proof. The implication from left to right 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 \( T(\Gamma; t) \) is defined and \( T(\Gamma; t) \subseteq \tau \).

- If \( \Gamma, x : \tau \vdash t : \tau \) then \( T(\Gamma, x : \tau; t) = \tau \).
- If \( \Gamma \vdash t : \tau' \) because of \( \Gamma \vdash t : \tau \) and \( \tau \subseteq \tau' \), then by the inductive hypothesis \( \Gamma \vdash t : T(\Gamma; t) \) and \( T(\Gamma; t) \subseteq \tau' \). Hence also \( T(\Gamma; t) \subseteq \tau' \) by Lemma C.12.
- Assume \( \Gamma \vdash ct_1 \ldots t_n : \rho^{s+1} \) because of \( \Gamma \vdash t_k : \sigma_k[\rho^s/A][\overline{\beta}/B] \) where \( \text{ArgTypes}(c_k) = (\sigma_1, \ldots, \sigma_n) \) and \( \text{Def}(c) = d \) and \( \rho = d(\overline{\tau}) \). Let \( \theta_k = T(\Gamma; t_k) \). By the inductive hypothesis \( \Gamma \vdash t_k : \theta_k \) and \( \theta_k \subseteq \sigma_k[\rho^s/A][\overline{\beta}/B] \). Hence by Lemma C.3 we have \( \theta_k = \sigma_k[\rho_k^s/A][\overline{\beta_k}/B] \) with \( \sigma_k \subseteq \sigma \) and \( \rho_k^s \subseteq \rho^s \) and \( \beta_k \subseteq \tau_j \) and \( \rho_k = d(\sigma_k) \) and \( \alpha_k \subseteq \tau_j \). We may assume \( s_k = 0 \) and \( \alpha_k = \bot \) if \( A \notin TV(\sigma_k) \) and \( \beta_k = \bot \) if \( B_j \notin TV(\sigma_k) \). Let \( \tau_j = \bigcup_{k=1}^m \alpha_k \sqcup \beta_k \).

Then \( T(\Gamma; ct_1 \ldots t_n) = d(\overline{\sigma(s+1)} \ldots \overline{\sigma(s+1)}) (\overline{\tau}) \) where \( m = \max \{ d \mid d \text{ is inductive, and } m = \min \text{ if } d \text{ is coinductive} \}. \) Together with Lemma C.14 we have \( \tau_j \subseteq \tau_i \). Together with properties of size expressions this implies \( T(\Gamma; ct_1 \ldots t_n) \subseteq \rho^{s+1} \).

- Assume \( \Gamma \vdash (\lambda x : \alpha. t) : \alpha \to \beta \) because of \( \Gamma, x : \alpha \vdash t : \beta \). By the inductive hypothesis \( \beta' = T(\Gamma, x : \alpha; t) \subseteq \beta \). Then \( T(\Gamma; (\lambda x : \alpha. t)) = T(\Gamma; t) \subseteq \beta \).

- Assume \( \Gamma \vdash t t' : \beta \) because of \( \Gamma \vdash t : \alpha \to \beta \) and \( \Gamma \vdash t' : \alpha \). By the inductive hypothesis \( T(\Gamma; t) \subseteq \alpha \to \beta \) and \( T(\Gamma; t') \subseteq \alpha \). Then \( T(\Gamma; t) \to T(\Gamma; t') \subseteq \alpha \to \beta \) and \( \alpha \subseteq \alpha' \) and \( \beta \subseteq \beta' \). We have \( T(\Gamma; t') \subseteq \alpha' \) by Lemma C.12. Hence \( T(\Gamma; t t') \subseteq \beta' \).

- Assume \( \Gamma \vdash ts : \tau[s/i] \) because of \( \Gamma \vdash t : \forall i. \tau \). By the inductive hypothesis \( T(\Gamma; t) \subseteq \forall \). Then \( T(\Gamma; t) = T(\Gamma; ts) \subseteq \tau[s/i] \).

- Assume \( \Gamma \vdash \text{case}(t; \{ c_k x_k \Rightarrow t_k \}) : \tau \) because of \( \Gamma \vdash t : \nu^{s+1} \) and \( \Gamma, x^1_k : \delta^1_k \vdash t_k : \tau \) and \( \text{ArgTypes}(c_k) = (\sigma^1_k, \ldots, \sigma^s_k) \) and \( \delta^1_k = \sigma^1_k[\nu^s/A][\overline{\beta}/B] \) and \( \nu = d(\overline{\beta}) \). By the inductive hypothesis \( T(\Gamma; t) \subseteq \nu^{s+1} \).

Then \( T(\Gamma; t) = d(\overline{\tau}) \) with \( s' \geq s + 1 \) and \( \beta' \subseteq \beta_k \). Let \( \gamma_k = \sigma^1_k[\delta^1_k(\overline{\tau})][\overline{\beta}/B] \). By Lemma C.8 we have \( \overline{\gamma} \geq s \). So by Lemma C.3 we have \( \gamma_k \subseteq \delta^1_k \). By the inductive hypothesis \( T(\Gamma, x^1_k : \delta^1_k, x^s_k : \delta^s_k, t_k) \subseteq \tau \). By Lemma C.16 we have \( T(\Gamma, x^1_k : \gamma^1_k, \ldots, x^s_k : \gamma^s_k, t_k) \subseteq T(\Gamma, x^1_k : \delta^1_k, \ldots, x^s_k : \delta^s_k, t_k), \) so \( T(\Gamma, x^1_k : \gamma^1_k, \ldots, x^s_k : \gamma^s_k, t_k) = t_k \subseteq \tau \). Let \( \tau' = \bigcup_{k=1}^m \tau_k \). Then \( T(\Gamma; \text{case}(t; \{ c_k x_k \Rightarrow t_k \})) = \tau' \).

By Corollary C.15 we have \( \tau' \subseteq \tau \).
• Other cases are analogous to the ones already considered or follow directly from the
inductive hypothesis.

C.2. Type checking. We now show that type checking in $\lambda^\circ$ 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 $j_1, \ldots, j_n$ with
$j_1 = i$ and $j_{k+1} \in SV(U(j_k))$ for $k = 1, \ldots, n - 1$, 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 $j \in SV(U(i))$. 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 $v : V_S \rightarrow \mathbb{N} \cup \{\omega\}$ such that $v'(s_1) > v'(s_2)$. Because $SV(s_1, s_2)$ is finite, there exist limit ordinals $v_1 < \ldots < v_n < \omega$ such that for each $i \in SV(s_1, s_2)$ either $v(i) = \infty$ or there are $k, m \in \mathbb{N}$
with $v(i) = v_k + m$. Let $M \in \mathbb{N}$ be maximal such that $v(i) = v_k + M$ for some $i, k$. Let $N$
be the maximal nesting of $+1$ in $s_1, s_2$, e.g., for a size expression $v(i + 1, j) + 1$ we have
$N = 2$. Let $j_k = k(M + N + 1)$ for $k = 1, \ldots, n$. Now it suffices to set $v'(i) = j_k + m$ if $v(i) = v_k + m$, and $v'(i) = \infty$ if $v(i) = \infty$.

**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.

**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 \( \infty \) or a size expression not containing \( \infty \). If \( U(i) = \infty \) for some \( i \in \text{dom}(U) \) then we may substitute \( \infty \) 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 \( \infty \) and for each \( (s_1, s_2) \in S' \) one of the following holds:

- neither \( s_1 \) nor \( s_2 \) contain \( \infty \),
- \( s_2 = \infty \) 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 \( \infty \) then \( (U', S') \) is not valid, because then \( v(s_2) < \infty \) for \( v : V \rightarrow \mathbb{N} \).

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

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

\[
i_1 = s_{i_1} \land \ldots \land i_k = s_{i_k} \land s_1 > = s_2 + 1
\]

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

Using the identities

\[
\max(a, b) + 1 = \max(a + 1, b + 1) \\
\min(a, b) + 1 = \min(a + 1, b + 1)
\]

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

\[
\min(a, b) \leq c \iff \exists n. (n \geq a \land n \geq b) \land n \leq c \\
c \leq \min(a, b) \iff \exists n. c \leq n \land n \leq a \land n \leq b \\
\max(a, b) \leq c \iff \exists n. a \leq n \land b \leq n \land n \leq c \\
c \leq \max(a, b) \iff \exists n. (n \leq a \land n \leq b) \land n \leq c
\]

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].

Lemma C.21. For any types \( \tau_1, \tau_2 \) there exists \( S = S(\tau_1, \tau_2) \) 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 \( \tau_1, \tau_2 \).

Proof. Follows by induction on the definition of \( \sqsubseteq \).

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


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. \( \Box \)

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

- \((U, S)\) is valid if \( T(\Gamma; t) \) is defined,
- if \( T(\Gamma; t) \) is defined then \( U(\tau) = T(\Gamma; t) \).

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

1. \( T'(U_0; \Gamma, x : \tau; x) = (U_0, \emptyset, \tau) \).
2. \( T'(U_0; \Gamma; c_1 \ldots t_n) = (U, S, \emptyset) \) if \( \theta = \mu^{\max(s_1, \ldots, s_n) + 1} \) and \( \text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_n) \) and \( \mu = d_\mu(\tau_1, \ldots, \tau_m) \) and \( \text{Def}(c) = d_\mu \) and \( T'(U_0; \Gamma; t_i) = (U_i, S_i, \theta_i) \) and
   \[
   \theta_i = \sigma_i[d_\mu(\alpha_1, \ldots, \alpha_n)/A][\beta_1/B_k, \ldots, \beta_n/B_m]
   \]
   (we take \( s_i = 0 \) and \( \alpha_j = \emptyset \) if \( A \notin \text{TV}(\sigma_i) \), and \( \beta_j = \emptyset \) if \( B_j \notin \text{TV}(\sigma_i) \); if some \( \theta_i \) does not have the desired form then the present case does not apply) and \( \tau_j = \bigcup_{i=1}^{n}(\alpha_j \sqcup \beta_j) \) (if \( \bigcup_{i=1}^{n}(\alpha_j \sqcup \beta_j) \) is not defined then the present case does not apply) and \( U = \bigcup_{i=0}^{n} U_i \) and \( S = \bigcup_{i=1}^{n} S_i \sqcup S(\sigma_i, \tau_i) \).
3. \( T'(U_0; \Gamma; c_1 \ldots t_n) = (U, S, \emptyset) \) if \( \theta = \nu^{\min(s_1, \ldots, s_n) + 1} \) and \( \text{ArgTypes}(c) = (\sigma_1, \ldots, \sigma_n) \) and \( \nu = d_\nu(\tau_1, \ldots, \tau_m) \) and \( \text{Def}(c) = d_\nu \) and \( T'(U_0; \Gamma; t_i) = (U_i, S_i, \theta_i) \) and
   \[
   \theta_i = \sigma_i[d_\nu(\alpha_1, \ldots, \alpha_n)/A][\beta_1/B_k, \ldots, \beta_n/B_m]
   \]
   (we take \( s_i = 0 \) and \( \alpha_j = \emptyset \) if \( A \notin \text{TV}(\sigma_i) \), and \( \beta_j = \emptyset \) if \( B_j \notin \text{TV}(\sigma_i) \); if some \( \theta_i \) does not have the desired form then the present case does not apply) and \( \tau_j = \bigcup_{i=1}^{n}(\alpha_j \sqcup \beta_j) \) (if \( \bigcup_{i=1}^{n}(\alpha_j \sqcup \beta_j) \) is not defined then the present case does not apply) and \( U = \bigcup_{i=0}^{n} U_i \) and \( S = \bigcup_{i=1}^{n} S_i \sqcup S(\sigma_i, \tau_i) \).
4. \( T'(U_0; \Gamma; x : \alpha \rightarrow \beta) = (U, S, \alpha \rightarrow \beta) \) if \( T'(U_0; \Gamma; x : \alpha ; t) = (U, S, \beta) \).
5. \( T'(U_0; \Gamma; tt') = (U, S, \beta) \) if \( T'(U_0; \Gamma; t) = (U_1, S_1, \alpha \rightarrow \beta) \) and \( T'(U_0; \Gamma; t') = (U_2, S_2, \alpha') \) and \( U = U_1 \cup U_2 \) and \( S = S_1 \cup S_2 \cup S(\alpha', \alpha) \).
6. \( T'(U_0; \Gamma; st) = (U \cup \{i = s\}, S, \tau) \) if \( T'(U_0; \Gamma; t) = (U, S, \forall i \cdot \tau) \) with \( i \) fresh.
7. \( T'(U_0; \Gamma; \lambda i. t) = (U, S, \forall i \cdot \tau) \) if \( T'(U_0; \Gamma; t) = (U, S, \tau) \) and \( i \notin \text{FSV}(\Gamma) \).
8. \( T'(U'; \Gamma; \text{case}(t; \{c_k \rightarrow x_k \mid k = 1, \ldots, n\})) = (U, S, \tau) \) if \( T'(U'; \Gamma; t) = (U_0, S_0, \sigma^s) \) and \( \mu = d(\beta) \) and \( \text{ArgTypes}(c_k) = (\sigma_k^a, \ldots, \sigma_k^{n_k}) \) and \( \delta^k_l = \sigma_k^l[\mu^l/A][\beta/B] \) with \( i \) fresh and \( T'(U' \cup \{i = s\}; \Gamma; x^k : \delta^k_1, \ldots, \delta^k_{n_k} \rightarrow x_k) = (U_k, S_k, \tau_k) \) and \( \tau = \bigcup_{k=1}^{n}(\tau_k) \) and \( U = \bigcup_{i=0}^{n} U_i \) and \( S = \bigcup_{i=0}^{n} S_i \).
9. \( T'(U'; \Gamma; \text{case}(t; \{c_k \rightarrow x_k \mid k = 1, \ldots, n\})) = (U, S, \tau) \) if \( T'(U'; \Gamma; t) = (U_0, S_0, \sigma^s) \) and \( \nu = d(\beta) \) and \( s \geq 1 \) and \( \text{ArgTypes}(c_k) = (\sigma_k^a, \ldots, \sigma_k^{n_k}) \) and \( \delta^k_l = \sigma_k^l[\nu^l/A][\beta/B] \) with \( i \) fresh and \( T'(U' \cup \{i = s\}; \Gamma; x^k : \delta^k_1, \ldots, \delta^k_{n_k} \rightarrow x_k) = (U_k, S_k, \tau_k) \) and \( \tau = \bigcup_{k=1}^{n}(\tau_k) \) and \( U = \bigcup_{i=0}^{n} U_i \) and \( S = \bigcup_{i=0}^{n} S_i \).
10. \( T'(U_0; \Gamma; \text{fix } f : \forall j_1 \ldots j_m. f \rightarrow \tau ; t) = (U, S, \forall j_1 \ldots j_m. \mu \rightarrow \tau) \) if \( T'(U_0; \Gamma, f : \forall j_1 \ldots j_m. \mu \rightarrow \tau; t) = (U, S, \theta) \) and \( i \notin \text{FSV}(\Gamma, \mu, \tau, j_1, \ldots, j_m) \) and \( S = S_0 \cup S(\theta, \forall j_1 \ldots j_m. \mu \rightarrow \tau) \).
11. \( T'(U_0; \Gamma; \text{cofix } f : \tau \rightarrow t) = (U, S, \tau) \) if \( T'(U_0; \Gamma, f : \text{chgtgt}(\tau, v^\min(s, j)); t) = (U, S, \theta) \)
and \(\text{tgt}(\tau) = \nu^s\) and \(j \notin \text{FSV}(\Gamma)\) and \(j \notin \text{SV}(\tau)\) and \(S = S_0 \cup S(\theta, \text{chgtgt}(\tau, \nu^{\min(s,j+1)}))\).

- Otherwise, if none of the above cases hold, we define \(T'(U_0; \Gamma; t) = (U_0, \{1 \leq 0\}, \bot)\) with \(\bot\) 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 \(\tau\) 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 size expression has size proportional to the size of a size context, so its size is at most \(O\) tree (i.e. when there are no more immediate recursive calls) the type \(\tau\) for any given call the result type is equal to at most the sum of sizes of \(\alpha\) sum of sizes of the types returned by immediate recursive calls (note that the size of \(\alpha\) is polynomial in \(\sum_{i} \mu\).

Indeed, there are essentially two possibilities of what we add to the context \(\Gamma\).

1. We add \(x : \alpha\) for the case of lambda abstraction \(\lambda x : \alpha.t'\). Then \(\alpha\) occurs in the original term \(t\), so the size of \(\Gamma\) grows by at most \(N\).

2. We add e.g. \(x_k^1 : \delta_{k}^{1}, \ldots, x_k^n : \delta_{k}^{nk}\) where \(\delta_{k}^{j} = \sigma_{k}^{j}[\mu/A][\beta/B]\) or the case of a case-term. Then \(\mu\) and \(\beta\) occur in the original term \(t\), and \(\sigma_{k}^{j}\) is an argument type for the constructor \(c_k\) which occurs in \(t\) (so the size of \(\sigma_{k}^{j}\) counts towards the size of \(t\)). Hence the total size of \(\sigma_{k}^{1}, \ldots, \sigma_{k}^{nk}\) is \(\leq N\), and thus so is the total number of occurrences of \(A, B\) in \(\sigma_{k}^{1}, \ldots, \sigma_{k}^{nk}\). The size of each of \(\mu, \beta\) is \(\leq N\). Therefore, the total size of \(\delta_{k}^{1}, \ldots, \delta_{k}^{nk}\) is at most \(N^2 + N\).

Hence, the size of the context at any given call to \(T'\) (during the whole run of the algorithm) is at most \(O(N^3)\). Let \((U, S, \tau)\) denote the result of calling \(T'\). At the leaves of the computation tree (i.e. when there are no more immediate recursive calls) the type \(\tau\) is taken from the context, so its size is at most \(O(N^3)\). At internal nodes, the size of \(\tau\) is equal to at most the sum of sizes of the types returned by immediate recursive calls (note that the size of \(\alpha \cup \beta\) is equal to at most the sum of sizes of \(\alpha\) and \(\beta\) 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 \(T'\) 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 \(\tau\) 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 \(S(\alpha, \beta)\) where each of \(\alpha, \beta\) is either a subtype of a type returned by an immediate recursive call or of the term \(t\). So the size of \(\alpha, \beta\) is polynomial in \(N\), and thus so is the size of \(S(\alpha, \beta)\) 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 \((U, S, \tau)\) has polynomial size. Note that in each of the calls to \(T'\), 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 $T'$). Hence, the whole running time is polynomial.

**Theorem 6.5.** Type checking in the system $\lambda^\omega$ is coNP-complete. More precisely, given $\Gamma, t, \tau$ 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 $\varphi$. For concreteness assume $\varphi$ is

$$(x \lor \neg y \lor z) \land (x \lor \neg z \lor y).$$

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

$$s_1 = \max(\min(x, \bar{y}, z) + 1, \min(x, \bar{z}, y) + 1, 1, \
\min(x, \bar{x}) + 1, \min(y, \bar{y}) + 1, \min(z, \bar{z}) + 1)$$

$$s_2 = \min(1, \max(x, \bar{x}), \max(y, \bar{y}), \max(z, \bar{z}))$$

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

Let $v$ with $\text{codom}(v) = \{\top, \bot\}$ be a satisfying valuation for $\varphi$. 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 $v(i) = \bot$, and $\bar{v}(i) = 1 - v(i)$, 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:

$$\min(x, \bar{y}, z) + 1 \leq 1 \land \min(x, \bar{y}, z) + 1 \leq \max(x, \bar{x}) \land \min(x, \bar{y}, z) + 1 \leq \max(y, \bar{y}) \land \min(x, \bar{y}, z) + 1 \leq \max(z, \bar{z}) \land \min(x, \bar{x}) + 1 \leq \max(z, \bar{z}) \land \min(y, \bar{y}) + 1 \leq 1 \land \min(y, \bar{y}) + 1 \leq \max(x, \bar{x}) \land \min(y, \bar{y}) + 1 \leq \max(x, \bar{x}) \land \min(y, \bar{y}) + 1 \leq \max(y, \bar{y}) \land \min(y, \bar{y}) + 1 \leq \max(z, \bar{z}) \land \min(y, \bar{y}) + 1 \leq \max(z, \bar{z}) \land \min(z, \bar{z}) + 1 \leq \max(x, \bar{x}) \land \min(z, \bar{z}) + 1 \leq \max(x, \bar{x}) \land \min(z, \bar{z}) + 1 \leq \max(y, \bar{y}) \land \min(z, \bar{z}) + 1 \leq \max(y, \bar{y}) \land \min(z, \bar{z}) + 1 \leq \max(z, \bar{z}).$$

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 $\text{codom}(v) = \{\top, \bot\}$ by $v(i) = \top$ if $\bar{v}(i) = 0$, and $v(i) = \bot$ 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 $\varphi$.

Hence for every 3-CNF boolean formula $\varphi$ there exists an equisatisfiable inequality $s_1 \leq s_2$ of size expressions which may be computed in polynomial time. So $\varphi$ 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 $\varphi$ is unsatisfiable iff $x : \nu s_1, f : \nu s_2 + 1 \rightarrow \mu^0 \vdash fx : \mu^0$ for some fixed $\nu, \mu$. □
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 $\Gamma = f : \forall i.\mu^i \to \mu^i$ and define $t_0 = f$, $t_{n+1} = \Delta_i.t_n \max(i, i)$. Let $s_0 = i$ and $s_{n+1} = \max(s_n, s_n)$. We have $T(\Gamma; t_n) = \forall i.\mu^{s_n} \to \mu^{s_n}$. The size of $s_n$ is proportional to $2^n$ while the size of $t_n$ is proportional to $n$.

Similarly, suppose $\mu$ has two constructors $c_1 : \mu \to \mu$ and $c_2 : \mu \to \mu$. Let $t_0 = x$ and $t_{n+1} = \text{case}(t_n; \{ c_1y \Rightarrow y, c_2y \Rightarrow y \})$. Let $s_n = 0 + 1 + 1 + \ldots + 1$ where 1 occurs $n$ times. By induction on $n$ one shows $T(x : \mu^{s_n}; t_n) = \mu^{s'_n}$ 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 $n$. 

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, 171 Second St, Suite 300, San Francisco, CA 94105, USA, or Eisenacher Strasse 2, 10777 Berlin, Germany