Towards Correctly Checking for Cycles in Overloaded Definitions

Arve Gengelbach

Uppsala University, Uppsala, Sweden

Johannes Åman Pohjola

University of New South Wales and CSIRO's Data61, Sydney, Australia

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2021-001
March 2021
ISSN 1404-3203

Towards Correctly Checking for Cycles in Overloaded Definitions

Arve Gengelbach
Uppsala University, Uppsala, Sweden

Johannes Åman Pohjola
University of New South Wales and CSIRO's Data61, Sydney, Australia

Abstract

Safe extension of a logic with definitional axioms need acyclic definitions, because cycles in definitions possibly entail contradiction. In this report we give a mechanised exact characterisation of acyclic overloading definitions. Our results support the soundness argument of the cyclicity checker of the Isabelle/HOL theorem prover, and serve as the theoretical foundation for future verification of such a cyclicity checker.

1 Introduction

Overloading definitions add complexity to a theorem prover. Adding new overloaded definitions to a proof development requires that each defined symbol does not depend on itself, as otherwise an extension by definitions with cyclic dependencies may render the resulting theory inconsistent. Acyclic dependencies (also called terminating dependencies) of definitions are a premise to consistent theories, as discussed in [KP19, ÅG20] and guarantee conservative definitions [GÅW21, GW20]. Hence, any theorem prover supporting overloading definitions should check that their dependencies are not cyclic.

Whether dependencies of definitions are cyclic is undecidable, because the post-correspondence problem can be embedded into a theory of overloaded definitions [Obu06]. Yet, Kunčar [Kun15] lays the theoretical foundation for checking cyclicity of *composable* dependencies which gives rise to a sound algorithm. Furthermore, Kunčar identifies *orthogonality* as a restriction on dependency relations that render the cyclicity checking algorithm complete and decidable.

This report extends the earlier work in two ways. First, in Section 3 we sketch the equivalence proof that composable dependencies are cyclic, if and only if there are cycles of certain shapes. In our proof we highlight the fixes of an innocent-looking false lemma, which relates the size of a type prior to and after non-bijective type variable instantiation. Our proofs in this section are mechanised in the HOL4 theorem prover as part of the CakeML project [TMK+19].[1] Second, in Section 4 we discuss why orthogonality of dependencies from overloading definitions is no suitable limitation.

As future work we aim to obtain a mechanised sound cyclicity checker, which composed with our earlier work [ÅG20, GÅW21] yields a verified theorem prover kernel that supports overloading.

2 Background

We generally follow the notation of Kunčar [Kun15].

2.1 Notation

In this section we define types and constant instances, that we call symbols, and different operands on these.

Types. We fix an infinite set TVar of type variables ranged over by α, β.

A *type signature* is a pair (K, arOf) of a finite set K of *type constructors*, where the arity of each type constructor is given by $\mathrm{arOf} : K \to \mathbb{N}_0$.

We further on assume a fixed signature and define the set of its types Type as the smallest set satisfying

- $\mathsf{TVar} \subseteq \mathsf{Type}$, and
- $(\sigma_1, \ldots, \sigma_{\mathrm{arOf}(k)})k \in \mathsf{Type}$, for $k \in K$ and $\sigma_1, \ldots, \sigma_{\mathrm{arOf}(k)} \in \mathsf{Type}$.

Types range over σ,τ.

For the discussion in Section 4 we assume that K contains a right-associative binary type constructor $\to$ for function types and a unary type constructor of Booleans bool. We identify bool with $()\mathsf{bool}$, and write function types as infix, like $\sigma \to \tau$ for $\sigma, \tau \in \mathsf{Type}$.

As inherited from [ÅG20, GÅW21], we define the size of a type recursively for type variables as $\mathsf{size}(\alpha) = 1$ and for types $(\sigma_1, \ldots, \sigma_{\mathrm{arOf}(\tau)})k \in \mathsf{Type}$ as

$$\mathsf{size}((\sigma_1, \ldots, \sigma_{\mathrm{arOf}(\tau)})k) = 1 + \mathrm{arOf}(\tau) + \sum_{i=1}^{\mathrm{arOf}(\tau)} \mathsf{size}(\sigma_i).$$

Type variables and unary types have the same size, e. g. $\mathsf{size}(\alpha) = \mathsf{size}(\mathsf{bool})$.

Note, Kunčar defines that type variables do not contribute to the size of a type (the base case is 0) and omits the arity $\mathrm{arOf}(\tau)$ in the second clause. This difference has no effect on the problems that we mention in Section 3.2.

The set $\mathrm{FV}(\tau)$ collects all type variables of its argument type τ.

[1] https://code.cakeml.org/tree/master/candle/overloading/syntax

Type instances. A *type substitution* is a function $\mathsf{TVar} \to \mathsf{Type}$ which replaces type variables by a type, and type substitutions homomorphically extend to type constructors. For a fixed type signature K, the set of all type substitutions is TSubst.

For any type substitution $\rho \in \mathsf{TSubst}$ and any type $\sigma \in \mathsf{Type}$, $\rho(\sigma)$ is a *(type) instance* of σ, written $\rho(\sigma) \leq \sigma$. For $\tau = \rho(\sigma)$ we annotate the type substitution ρ as in $\tau \leq_\rho \sigma$. For easier reading, $\geq$ is $\leq$ with flipped arguments. Two types σ and τ are *orthogonal* if they have no common type instance, i. e. if for all $\rho, \rho' \in \mathsf{TSubst}$, holds $\rho(\sigma) \neq \rho'(\tau)$.

A type substitution is a *(variable) renaming* if it is a permutation on a subset of TVar. Hence, a renaming η keeps the structure of a type τ, and has no effect on the size, $\mathsf{size}(\eta(\tau)) = \mathsf{size}(\tau)$, nor on the number of type variables $\#\,\mathrm{FV}(\eta(\tau)) = \#\,\mathrm{FV}(\tau)$.

Equivalences. Two type substitutions $\rho, \rho' \in \mathsf{TSubst}$ are equal on a type τ, written $\rho =_\tau \rho'$, if $\rho(\tau) = \rho'(\tau)$. This equality holds point-wise at every of the type variables of τ. Two type substitutions ρ and ρ' are *equivalent*, $\rho \approx \rho'$ if there exists a renaming η such that $\rho = \eta \circ \rho'$. We write $\rho \approx_\tau \rho'$ if the equality holds on a type τ only: $\rho =_\tau \eta \circ \rho'$.

For example, any renaming η is equivalent to the identity $\eta \approx \mathsf{id}$, as witnessed by the inverse η^{-1}.

Lifting to constants. We extend the type signature by a set of constants Const, and by a function $\mathrm{tpOf} : \mathsf{Const} \to \mathsf{Type}$ that assigns each constant symbol its type. The set of constant instances CInst contains all tuples (c, τ) written c_τ, whose type is an instance of the type of c, $\tau \leq \mathrm{tpOf}(c)$.

We define $\mathsf{Symb} := \mathsf{CInst} \cup \mathsf{Type}$ and lift the notions defined for types to constant instances canonically. For example, we define $\mathsf{size}(c_\sigma) = \mathsf{size}(\sigma)$, and $\rho(c_\sigma) = c_{\rho(\sigma)}$, and $c_\sigma \leq d_\tau$ iff $c = d$ and $\sigma \leq \tau$. We call the elements from Symb as *symbols* and let them range over p, q, r, s.

We define *orthogonality* $p \,\#\, q$ for pairs (p, q) of types and constant instances that either one of p and q is a type and the other a constant instance, or otherwise both are orthogonal $p \,\#\, q$.

2.2 Dependency Relation

For relations $\mathcal{R}$, $\mathcal{R}^+$ is the transitive closure and $\mathcal{R}^*$ is the reflexive-transitive closure. For a binary relation of symbols we write $\rightsquigarrow \subseteq \mathsf{Symb} \times \mathsf{Symb}$, and say *dependency relation.*

The *type-substitutive closure* $\rightsquigarrow^{\downarrow}$ of a dependency relation $\rightsquigarrow$ on symbols is the smallest relation such that $\rho \in \mathsf{TSubst}$ and $x \rightsquigarrow y$ implies $\rho(x) \rightsquigarrow^{\downarrow} \rho(y)$.

A dependency relation $\rightsquigarrow$ is *monotone*, if for every $p \rightsquigarrow q$, we have $\mathrm{FV}(q) \subseteq \mathrm{FV}(p)$, i. e. all type variables of q are bound through p. Monotonicity holds for dependencies of type and constant definitions as introduced in [KP19].

A dependency relation $\rightsquigarrow$ is *cyclic* (also *non-terminating*) if there exists an infinite sequence of type substitutions ρ_i and $p_i \rightsquigarrow q_i$ such that $\rho_i(q_i) = \rho_{i+1}(p_{i+1})$ and $p_i \rightsquigarrow q_i$ for all $i \in \mathbb{N}_0$. The constraint of these chains of dependencies stem from that an instance $\rho_i(p_i)$ of a defined or declared symbol p_i is defined in terms of the symbol p_{i+1}. Thus, to evaluate the instance $\rho_i(p_i)$ one needs to evaluate its dependency $\rho_{i+1}(p_{i+1})$.

Composability. A dependency relation $\rightsquigarrow$ on Symb is *composable*, if for any sequence $(r_i)_{0 \leq i \leq n}$ of symbols with $r_0 \rightsquigarrow r_1$ and for $1 \leq i < n$ with $r_i \rightsquigarrow^{\downarrow} r_{i+1}$ for any possible continuation $p \rightsquigarrow q$ holds $r_n \leq p$, or $r_n \geq p$, or otherwise r_n and p are orthogonal, $r_n \mathbin{\#} p$. That means that the fourth case $\rho(r_n) = \rho'(p)$ for some non-trivial $\rho, \rho' \in \mathsf{TSubst}$ does not occur.

2.3 Solutions

In this section we will define transitive dependency chains in $\rightsquigarrow^{\downarrow}$ as solutions, and prove basic properties of most general solutions.

For a finite sequence of dependencies $(p_i, q_i)_{i \leq n}$, i. e. $p_i \rightsquigarrow q_i$ for $0 \leq i \leq n$, a sequence of type substitutions $(\rho_i)_{i \leq n}$ is a *solution* if for every $0 \leq i < n$ it holds $\rho_i(q_i) = \rho_{i+1}(p_{i+1})$. An infinite sequence of type substitutions $(\rho_i)_{i \in \mathbb{N}_0}$ is a solution of $(p_i, q_i)_{i \in \mathbb{N}_0}$, if every finite prefix is.

In finding a solution, the *most general* solutions are interesting, i. e. any other solution can be obtained as a type instance of a most general solution. Most general solutions are unique up-to invertible renamings of type variables ([Kun15, Lemma 5.4]). For a monotone relation we obtain the following formulation:

Lemma 1. *For a monotone dependency relation $\rightsquigarrow$, and for two most general solutions $(\rho_i)_{i \leq n}$ and $(\rho'_i)_{i \leq n}$ of $(p_i, q_i)_{i \leq n} \subseteq \rightsquigarrow$ there exists an invertible variable renaming η such that $\rho_i = (\eta \circ \rho'_i)(p_i)$ for all $0 \leq i \leq n$.*

By monotonicity, solutions are most general if the type substitution at index 0 is equivalent to the identity.

Lemma 2. *For a monotone relation $\rightsquigarrow$, any solution $(\rho_i)_{i \leq n}$ to $(p_i, q_i)_{i \leq n} \subseteq \rightsquigarrow$ with $\rho_0 \approx_{p_0} \mathsf{id}$ is a most general solution.*

Proof. Assume that ρ_0 acts as the identity on type variables not from $\mathrm{FV}(p_0)$ Furthermore, we assume that $\rho_0 =_{p_0} \mathsf{id}$. (By assumption ρ_0 is invertible, thus as the dependency relation is monotone, if one of the sequences $(\rho_0^{-1} \circ \rho_i)_{i \leq n}$ and $(\rho_i)_{i \leq n}$ is a most general solution, so is the other.) Let $(\rho'_i)_{i \leq n}$ be another solution to $(p_i, q_i)_{i \leq n}$, then by induction we show that $\rho'_0 \circ \rho_i =_{p_i} \rho'_i$ for all $i \leq n$. At $i = 0$, we have $\rho'_0 \circ \rho_0 =_{p_0} \rho'_0$. For $i > 0$, for all $j < i$ it holds $\rho'_0 \circ \rho_j =_{p_j} \rho'_j$, and by monotonicity $\mathrm{FV}(q_j) \subseteq \mathrm{FV}(p_j)$ this equality also holds on q_i, $\rho'_0 \circ \rho_j =_{q_j} \rho'_j$. The claim $\rho'_0 \circ \rho_i =_{p_i} \rho'_i$ follows, because both $(\rho_i)_{i \leq n}$ and $(\rho'_i)_{i \leq n}$ are solutions:

$$(\rho'_0 \circ \rho_i)(p_i) = (\rho'_0 \circ \rho_{i-1})(q_{i-1}) = \rho'_{i-1}(q_{i-1}) = \rho'_i(p_i)$$

□

3 Equivalence of Acyclicity

In this section we discuss the equivalence proof that permits a different formulation of cyclic dependencies from overloading definitions. The argument closely follows the rigorous pen-and-paper proof of Kunčar [Kun15].

We first state in Section 3.1 the problem that we aim to solve, and discuss in Section 3.2 why we slightly deviate from Kunčar's proof. In Section 3.3 we discuss properties of extending solutions by one step and conclude this part with a proof of the key lemma in Section 3.4.

3.1 Equivalence of Acyclicity

The key insight is that composable dependency relations allow a different characterisation of the acyclicity problem. The main theorem [Kun15, Lemma 5.17] states that for a finite, monotone and composable dependency relation $\rightsquigarrow$ the following equivalence holds:

Theorem 3. *The type-substitutive, transitive closure* $\rightsquigarrow^{\downarrow+}$ *of the dependency relation is cyclic iff the type-substitutive closure has a cycle of the form* $x \rightsquigarrow y \rightsquigarrow^{\downarrow*} \rho(x)$ *for some* $\rho \in \mathsf{TSubst}$*, and some* $x, y \in \mathsf{Symb}$.

In other words, cyclic dependencies are discoverable by searching for cycles of the latter shape. The discovered cycles are infinite solutions whose type substitution at index 0 is the identity id. Essential to an algorithmic search is, that the first step is not type-substitutive.

3.2 The Measure

The equivalence follows through a contradiction by infinite descent as discussed in Section 3.4. For cyclic dependencies we will achieve that the change in number of free variables strictly decreases or the change in type size strictly increases.

Lemma 3.1b in [Kun15] misses that a non-bijective type substitution may leave the size of a type unchanged. For example, instantiating the polymorphic function type $\alpha \to \beta$ with $\rho = \alpha \mapsto \beta$ results in $\rho(\alpha \to \beta) = \beta \to \beta$, where ρ does not affect the size. We correct the statement as follows:

Lemma 4. *For two symbols p and q with $q \geq_\rho p$, holds* $\mathsf{size}(q) \leq \mathsf{size}(p)$. *If additionally $q \not\leq p$ holds, then* $\#\mathrm{FV}(p) < \#\mathrm{FV}(q)$ *or* $\mathsf{size}(q) < \mathsf{size}(p)$.

In other words, applying the type substitution ρ to q (observe that $p = \rho(q)$) may increase the size $\mathsf{size}(q) \leq \mathsf{size}(\rho(q))$. In the second part of Lemma 4, *q is no instance of p*, written $q \not\lesssim p$, means that for all $\rho' \in \mathsf{TSubst}$ the symbol q is $q \neq \rho'(p)$. Thus ρ applied to q is not invertible (Kunčar describes this as $\rho \not\approx_q \mathsf{id}$) and applying ρ to q may unify at least two type variables of $\mathrm{FV}(q)$ or strictly increase the size. In the latter case ρ instantiates at least one of the type variables of q with a type constructor.

This lemma entails adjustments to the proofs, that we mainly discuss in Sections 3.3 and 3.4.

3.3 Extending Solutions by One Step

Due to composability the solutions that can be extended by one dependency step have particular shapes and properties, that we discuss in this section.

By composability, trying to extend a solution $(\rho_i)_{i \leq n}$ of $(p_i, q_i)_{i \leq n}$ (with initial $\rho_0 = \mathsf{id}$) by one step gives three cases: either for all $p \rightsquigarrow q$ orthogonality holds $\rho_n(q_n) \# p$ and no extension is possible, or an extension is possible as there exists $p \rightsquigarrow q$ such that $\rho_n(q_n) \leq p$, or $\rho_n(q_n) \geq p$.

3.3.1 $\leq$-extension

If a solution is extendable with a $\leq$ step (i. e. $\rho_n(q_n) \leq p$), the resulting solution has the following shape [Kun15, Lemma 5.9].

Lemma 5. *For a most general solution $(\rho_i)_{i \leq n}$ of a family of pairs $(p_i, q_i)_{i \leq n}$ from a monotone dependency relation with $p_{n+1} \rightsquigarrow q_{n+1}$ and $\rho_n(q_n) \leq_{\rho'} p_{n+1}$, the following is a most general solution to the longer sequence $(p_i, q_i)_{i \leq n+1}$:*

$$(\rho_i)_{i \leq n}, \rho'.$$

The previous Lemma 5 entails the following invariant of the size and number of type variables in a $\leq$-extension of a solution.

Lemma 6. *For a monotone dependency relation $\rightsquigarrow$, let $(p_i, q_i)_{i \leq n+1} \subseteq \rightsquigarrow$ be dependency pairs with a most general solution $(\rho_i)_{i \leq n}$ of the shorter sequence and with $(\rho'_i)_{i \leq n+1}$ a most general solution of the longer sequence.*
If $\rho_n(q_n) \leq_{\rho'} p_{n+1}$, then the following two equalities hold:

$$\mathsf{size}(\rho_0(p_0)) = \mathsf{size}(\rho'_0(p_0)) \qquad \# \mathrm{FV}(\rho_0(p_0)) = \# \mathrm{FV}(\rho'_0(p_0)).$$

Observing that by Lemma 1 the most general solutions $(\rho_i)_{i \leq n}, \rho'$ and $(\rho'_i)_{i \leq n+1}$ are equivalent on p_0 modulo renaming of type variables and observing that renamings have no effect on the type size and number of type variables, the proof of Lemma 6 is easy. As a consequence, a sequence of consecutive $\leq$-extensions has no effect on the size nor the number of free variables at p_0 w. r. t. the most general solutions at index 0.

3.3.2 (Strict) $\geq$-extension

Analogously to the previous Section 3.3.1, we look at the other possible way to extend a solution and the change of the size of the type or the number of type variables. If a solution is extendable with a $\geq$ step (i. e. $\rho_n(q_n) \geq p$), the resulting solution has the following shape [Kun15, Lemma 5.10].

Lemma 7. *For a most general solution $(\rho_i)_{i \leq n}$ of a family of pairs $(p_i, q_i)_{i \leq n}$ from a monotone dependency relation with $p_{n+1} \rightsquigarrow q_{n+1}$ and $\rho_n(q_n) \geq_{\rho'} p_{n+1}$, there exists a type substitution $\hat{\rho} \in \mathsf{TSubst}$ such that $\hat{\rho} =_{\rho_n(q_n)} \rho'$, $\hat{\rho}(\alpha) = \alpha$ for all $\alpha \in \mathrm{FV}(\rho_0(p_0)) \setminus \mathrm{FV}(\rho_n(q_n))$ and the following is a most general solution of the family $(p_i, q_i)_{i \leq n+1}$:*

$$(\hat{\rho} \circ \rho_i)_{i \leq n}, \mathsf{id}\,.$$

As Lemma 6 already covers the case when in an extension step both of $\rho_n(q_n) \leq p_{n+1}$ and $\rho_n(q_n) \geq p_{n+1}$ hold, we regard the effect of strict $\geq$-extension. With a strict $\geq$-extension of a solution $(\rho_i)_{i \leq n}$ we mean that the following is true:

$$\rho_n(q_n) \not\leq p_{n+1} \qquad\qquad \text{and} \qquad\qquad \rho_n(q_n) \geq p_{n+1}.$$

The initial Lemma 4 states a property that applies in this situation, and which we can port from index n to index 0 by monotonicity.

Lemma 8. *For a monotone dependency relation $\rightsquigarrow$, let $(p_i, q_i)_{i \leq n+1} \subseteq \rightsquigarrow$ be dependency pairs with a most general solution $(\rho_i)_{i \leq n}$ of the shorter sequence and with $(\rho'_i)_{i \leq n+1}$ a most general solution of the full sequence.*

If both $\rho_n(q_n) \not\leq p_{n+1}$ and $\rho_n(q_n) \geq_{\rho'} p_{n+1}$ hold, then we can derive the inequality $\mathsf{size}(\rho_0(p_0)) \leq \mathsf{size}(\rho'_0(p_0))$, and additionally the formula:

$$\mathsf{size}(\rho_0(p_0)) < \mathsf{size}(\rho'_0(p_0)) \vee \#\,\mathrm{FV}(\rho'_0(p_0)) < \#\,\mathrm{FV}(\rho_0(p_0)).$$

Proof. For the most general solution $(\rho_i)_{i \leq n}$ with $\rho_n(q_n) \geq_{\rho'} p_{n+1}$, let ρ' be as obtained by Lemma 7. Thus, all type variables from the two sets $R := \mathrm{FV}(\rho_0(p_0)) \setminus \mathrm{FV}(\rho_n(q_n))$ and $C := \mathrm{FV}(\rho'(\rho_n(q_n)))$ are named apart. The type variables in R are those variables that got instantiated at different extension steps between 0 and n. We may assume that ρ' acts as the identity everywhere else except on $\mathrm{FV}(\rho_n(q_n))$.

By Lemma 1 we can assume that $\rho' \circ \rho_i =_{p_i} \rho'_i$ for $i \leq n$.

Assume $\mathsf{size}(\rho_0(p_0)) = \mathsf{size}((\rho' \circ \rho_0)(p_0))$. The following claim remains:

$$\#\,\mathrm{FV}((\rho' \circ \rho_0)(p_0)) < \#\,\mathrm{FV}(\rho_0(p_0)).$$

By the definition of size, the equality entails that ρ' only renames variables and instantiates type variables by type constructors of arity 0, and unifies distinct type variables, thus for any p it holds (1): $\#\,\mathrm{FV}(\rho'(p)) = \#(\rho'(\mathrm{FV}(p)))$.

By monotonicity holds $\mathrm{FV}(\rho_n(q_n)) \subseteq \mathrm{FV}(\rho_0(p_0))$, and we can regard the disjoint partition (2): $\mathrm{FV}(\rho_0(p_0)) = R \cup \mathrm{FV}(\rho_n(q_n))$. For the type substitution ρ' it holds, (3) that $\rho'(R) \cap C = \emptyset$ and on the type variables R, the substitution acts as the identity (4) $\rho'(R) = R$. We obtain the following equality.

$$\begin{aligned} \#\,\mathrm{FV}((\rho' \circ \rho_0)(p_0)) &\overset{(1)}{=} \#\left(\rho'\left(\mathrm{FV}(\rho_0(p_0))\right)\right) \overset{(2)}{=} \#\left(\rho'(R) \cup \rho'(\mathrm{FV}(\rho_n(p_n)))\right) \\ &\overset{(1)}{=} \#\left(\rho'(R) \cup C\right) \overset{(3)}{=} \#\rho'(R) + \#C \overset{(4)}{=} \#R + \#C \end{aligned}$$

By (2) we rewrite the right-hand side in the remaining claim:

$$\#\,\mathrm{FV}((\rho' \circ \rho_0)(p_0)) < \#R + \#\,\mathrm{FV}\left(\rho_n\left(q_n\right)\right)$$

Hence, it remains to prove $\#C < \#\,\mathrm{FV}(\rho_n(q_n))$ for $C = \mathrm{FV}(\rho'(\rho_n(q_n)))$. But this strict inequality holds by Lemma 4: The equality of sizes

$$\mathsf{size}(\rho_n(p_n)) = \mathsf{size}((\rho' \circ \rho_n)(p_n))$$

holds, as it holds on the superset $\mathrm{FV}(\rho_0(p_0)) \supseteq \mathrm{FV}(\rho_n(p_n))$, which proves the lemma. □

Consequently a strict $\geq$-extension of a solution entails, that a solution at index 0 unifies at least two type variables or instantiates at least one type variable with a type constructor.

3.4 Proving *The Key Technical Lemma*

Composability implies that an infinite chain of dependencies contains only finitely many strict $\geq$-extensions, which we prove in this section.

We define an infinite solution as *k-ascending* if from index k onward only $\leq$-extensions occur.

Definition 9. *For a family $(p_i, q_i)_{i \leq n}$ and $p \in \mathsf{Symb}$ we write $(p_i, q_i)_{i \leq n} \preccurlyeq p$ if there exists a most general solution $(\rho_i)_{i \leq n}$ such that $\rho_n(q_n) \leq p$.*
*An infinite family $(p_i, q_i)_{i \in \mathbb{N}_0}$ is k-*ascending*, if for all $n \geq k$ it holds that $(p_i, q_i)_{i \leq n} \preccurlyeq p_{n+1}$.*

As the following Lemma 10 proves, a suffix of a k-ascending sequence is 0-ascending, because at any $\geq$-extension step the solution is id.

Lemma 10. *Let $\rightsquigarrow$ be a composable, monotone dependency relation and $(p_i, q_i)_{i \in \mathbb{N}_0}$ an infinite sequence of dependency pairs. If $(p_i, q_i)_{i \in \mathbb{N}_0}$ is k-ascending then $(p_i, q_i)_{k \leq i}$ is 0-ascending.*

Proof. Let $0 < k$ be the smallest, such that $(p_i, q_i)_{i \in \mathbb{N}_0}$ is k-ascending and let $(\rho_i)_{i \leq k-1}$ be a most general solution. As k is smallest, it holds $\rho_{k-1}(q_{k-1}) \not\leq$

p_k and hence $\rho_n(q_{k-1}) \geq p_k$, by composability. By Lemma 7, there exists a $\hat{\rho}$ such that $(\hat{\rho} \circ \rho_i)_{i \leq k-1}, \mathsf{id}$ is a most general solution to $(p_i, q_i)_{i \leq k}$. Consequently by Lemma 1, any most general solution $(\rho'_i)_{i \leq k}$ of $(p_i, q_i)_{i \leq k}$ is at index k equivalent to the identity: $\rho'_k \approx_{p_k} \mathsf{id}$.

For any $k' \geq k$, we show that if $(\rho_i)_{i \leq k'}$ is a most general solution for $(p_i, q_i)_{i \leq k'}$, then the suffix $(\rho_i)_{k \leq i \leq k'}$ is a most general solution for $(p_i, q_i)_{k \leq i \leq k'}$. As all the steps from k to k' are $\leq$-extensions, thus $\rho_k \approx_{p_k} \mathsf{id}$. Then $(\rho_i)_{k \leq i \leq k'}$ is a most general solution for $(p_i, q_i)_{k \leq i \leq k'}$, by Lemma 2. □

With these preparations, we prove the key lemma [Kun15, Lem 5.16].

Theorem 11 (The Key Technical Lemma)**.** *For a composable and monotone dependency relation $\rightsquigarrow$ and $(p_i, q_i)_{i \in \mathbb{N}_0} \subseteq \rightsquigarrow$ an infinite sequence of dependency pairs, such that every finite prefix of $(p_i, q_i)_{i \in \mathbb{N}_0}$ has a solution, then there exists an index k with $(p_i, q_i)_{k \leq i}$ is 0-ascending.*

Proof. By Lemma 10 it suffices to show that there exists an index k such that $(p_i, q_i)_{i \in \mathbb{N}_0}$ is k-ascending, i. e. from index k onward only $\leq$-extensions occur. Assume the contrary, that for each k there exists a smallest $k' > k$ such that if $(\rho_i)_{i<k'}$ is a most general solution for $(p_i, q_i)_{i<k'}$ then $\rho_{k'-1}(q_{k'-1}) \not\leq p_{k'}$. By composability also holds $\rho_{k'-1}(q_{k'-1}) \geq p_{k'}$.

Denote by $(k_j)_{j \in \mathbb{N}_0}$ the sequence of all indices that iterate all the strict $\geq$-extensions. That is, for any n, any j and any most general solution $(\rho_i)_{i \leq n}$ for $(p_i, q_i)_{i \leq n}$, we have:

For $k_j < n < k_{j+1}$ we have a $\leq$-extension $\rho_n(q_n) \leq p_{n+1}$, and otherwise, for $n = k_j$ we have a strict $\geq$-extension $\rho_n(q_n) \not\leq p_{n+1}$ and $\rho_n(q_n) \geq p_{n+1}$.

For $n \in \mathbb{N}_0$, we consider each of the two cases $k_j < n < k_{j+1}$ and $n = k_j$ individually, and denote by $(\rho_i^{(j)})_{i \leq j}$ a most general solution of $(p_i, q_i)_{i \leq j}$. It is irrelevant which representant of the equivalence class of most general solutions of length j (modulo bijective renaming of type variables) is chosen, because size and number of free type variables are invariant under renaming of type variables. Furthermore, we abbreviate $r^{(i)} = \rho_0^{(i)}(p_0)$.

$\leq$-extension: Assume $k_j + 1 < k_{j+1}$, i. e. there is at least one $\leq$-extension. Then for all m such that $k_j < n \leq m \leq k_{j+1}$ we obtain by induction and by Lemma 6:

$$\mathsf{size}(r^{(n)}) = \mathsf{size}(r^{(m)}) \qquad\qquad \#\,\mathrm{FV}(r^{(n)}) = \#\,\mathrm{FV}(r^{(m)}).$$

Strict $\geq$-extension: Otherwise, we regard the case $n = k_j$. By Lemma 8, we obtain $\mathsf{size}(r^{(n)}) \leq \mathsf{size}(r^{(n+1)})$, and

$$\mathsf{size}(r^{(n)}) < \mathsf{size}(r^{(n+1)}) \vee \#\,\mathrm{FV}(r^{(n+1)}) < \#\,\mathrm{FV}(r^{(n)}).$$

Thus at $\leq$-extension steps equality holds and at strict $\geq$-extension steps, at index 0 the size increases strictly or the number of type variables decreases strictly. Assume there is an index l, such that for all $i \geq l$, the size at index 0 is constant, $\mathsf{size}(r^{(l)}) = \mathsf{size}(r^{(i)})$. Combining both of the properties of $(k_j)_{j \in \mathbb{N}_0}$, a contradiction follows, as for each of the infinitely many indices $k_j > l$ it holds:

$$\#\,\mathrm{FV}(r^{(k_{j+1})}) < \#\,\mathrm{FV}(r^{(k_j)})$$

Thus, there are infinitely many indices $I \subseteq (k_j)_{j \in \mathbb{N}_0}$ such that for all $n \in I$:

$$\mathsf{size}(r^{(n)}) < \mathsf{size}(r^{(n+1)})$$

Let $(\rho_i)_{i \in \mathbb{N}_0}$ be a solution for $(p_i, q_i)_{i \in \mathbb{N}_0}$. As the size grows infinitely, there exists an index $n \in \mathbb{N}_0$ such that the size of $r^{(n)} = \rho_0^{(n)}(p_0)$ has grown larger than the size of $\rho_0(p_0)$, i. e. $\mathsf{size}(\rho_0(p_0)) < \mathsf{size}(r^{(n)})$. As $(\rho_i^{(n)})_{i \leq n}$ is a most general solution for $(p_i, q_i)_{i \leq n}$, there exist type substitutions $(\eta_i)_{i \leq n}$ that witness that $(\rho_i)_{i \leq n}$ is an instance: $(\eta_i \circ \rho_i^{(n)})(p_i) = \rho_i(p_i)$, for all $i \leq n$. Because any type substitution increases the size of a type, $\mathsf{size}(r^{(n)}) \leq \mathsf{size}(\eta_0(r^{(n)}))$, we obtain the contradiction:

$$\mathsf{size}(r^{(n)}) \leq \mathsf{size}(\eta_0(r^{(n)})) = \mathsf{size}(\rho_0(p_0)) < \mathsf{size}(r^{(n)})$$

This proves the claim. □

As a conclusion, for monotone, composable dependency relations $\rightsquigarrow$, cycles in the type-substitutive hull $\rightsquigarrow^{\downarrow}$ start with a non-type-substitutive step. Theorem 3 follows, as the proof of [Kun15, Lemma 5.17], holds unchanged as our formalisation certifies. We mechanised the proofs for a specific dependency relation which originates from dependencies of overloading definitions [ÅG20].

4 Cyclicity of Overloading Definitions

In this section we discuss how the proven equivalence of cyclic dependencies (cf. Theorem 3) can be used in a cyclicity checker for theories of type definitions and overloading constant definitions. In the discussion we also focus on the sufficient properties that render a dependency relation decidable, according to Kunčar [Kun15].

4.1 Algorithmically Checking for Cycles

A verified theorem prover would need to check if a theory is definitional, i. e. finite, consisting of pair-wise orthogonal definitions and the closure of its dependency relation is terminating. For finitely many definitions the induced dependency relation is finite, orthogonality can be determined in polynomial time and monotonicity holds by definition.

The search for cycles in the dependency relation $\rightsquigarrow$ of the theory, amounts to a breadth-first search of the type-substitutive closure $\rightsquigarrow^{\downarrow}$, from each of the finitely many tuples $p \rightsquigarrow q$. At the i-th iteration step,

$$p \rightsquigarrow p^{(1)} \rightsquigarrow^{\downarrow} \cdots \rightsquigarrow^{\downarrow} p^{(i)},$$

this path through the type-substitutive reflexive hull $\rightsquigarrow^{\downarrow *}$ can be checked for composability (whether for each $x \rightsquigarrow y$ one of the instantiations $p^{(i)} \leq x$, or $p^{(i)} \geq x$, or orthogonality $p^{(i)} \# x$ holds). Following a successful check, the path $p \rightsquigarrow^{\downarrow +} p^{(i)}$ can be extended for each $x \rightsquigarrow y$ with $p^{(i)} \leq_{\rho} x$, which means $p^{(i)} = \rho(x)$. By Theorem 11, the other case $p^{(i)} \geq x$ is irrelevant for finding cycles. However, if $x = p$ a cycle $p \rightsquigarrow^{\downarrow +} \rho(p)$ is found. Otherwise, $p^{(i)} \rightsquigarrow^{\downarrow} \rho(y)$, and the search continues with $p^{(i+1)} = \rho(y)$.

The proven equivalence of Theorem 3 [Kun15, Lemma 5.17] of cyclic dependency relations holds, as we have motivated above and shown in a mechanised proof. That implies that the described algorithm is sound.

Despite its unclear termination without further restrictions on the dependency relation, this algorithm is close the algorithm used in Isabelle/HOL.[2] For the purpose of checking orthogonality and whether a type is an instance of another type, we have formalised a unification algorithm (based on [BSN+01]) and an instantiation check, each of which we have proven sound and complete in a mechanised proof.

4.2 Restricting Dependencies for Termination

With further restriction of (1) and (2) (cf. [Kun15, Def. 6.1]) on the relation, Kunčar discovered that the iteration terminates.

$$\forall p\, q\, p'\, q'.\, p \rightsquigarrow q \wedge p' \rightsquigarrow q' \wedge p \neq p' \implies p \# p' \tag{1}$$
$$\forall p\, q\, p'\, q'.\, p \rightsquigarrow q \wedge p' \rightsquigarrow q' \wedge q \neq q' \implies p \# p' \tag{2}$$

We exemplify that each of these restrictions are not satisfiable for dependency relations induced by theories of definitions.

First, to compensate for a larger dependency relation, by [ÅG20], the dependencies in the first conjunct (1) needs to exclude any dependencies arising from declarations. (Note, that [ÅG20] was published well after Kunčar's work.) A theory with a polymorphic constant $\mathsf{size}_{\alpha \to \mathbb{N}_0}$ and a definition of a proper instance $\mathsf{size}_{\alpha\ \mathsf{list} \to \mathbb{N}_0}$, induces among others the following dependencies, where the latter originates from the constant declaration of $\mathsf{size}_{\alpha \to \mathbb{N}_0}$.

$$\mathsf{size}_{\alpha\ \mathsf{list} \to \mathbb{N}_0} \rightsquigarrow \alpha\ \mathsf{list} \qquad\qquad \text{and} \qquad\qquad \mathsf{size}_{\alpha} \rightsquigarrow \alpha$$

This is an example that contradicts (1), as $\mathsf{size}_{\alpha\ \mathsf{list} \to \mathbb{N}_0} \neq \mathsf{size}_{\alpha}$, but both constant instances are not orthogonal.

[2] cf. https://isabelle.sketis.net/repos/isabelle/file/Isabelle2020/src/Pure/defs.ML#181

Second, the other conjunct (2) is too restrictive, even for dependency relations that are induced by theories of definitions without overloaded constants. In the following a constant ONTO of type $(\alpha \to \beta) \to \mathsf{bool}$ is defined by a lambda expression with quantifier constants.

$$\mathsf{ONTO}_{(\alpha\to\beta)\to\mathsf{bool}} \equiv \lambda f.\ \forall y_\beta.\ \exists x_\alpha.\ y = f(x)$$

The formula $\mathsf{ONTO}\, f$ expresses that $f_{\alpha\to\beta}$ is a surjective function, which allows to formulate the axiom of Dedekind-infinity in [ÅG20]. Among others, this constant definition introduces the following dependencies.

$$\mathsf{ONTO} \rightsquigarrow \forall \rightsquigarrow^{\downarrow} \alpha \qquad\qquad \text{and} \qquad\qquad \mathsf{ONTO} \rightsquigarrow \exists \rightsquigarrow \alpha$$

These dependencies violate the restriction of (2). In (2) any branching dependency from $p = p' = \mathsf{ONTO}$ is not orthogonal; for any p holds $\neg(p \,\#\, p)$. Furthermore, as a consequence of the amendments to the dependency relation in [ÅG20] the given chains of dependencies do not satisfy [Kun15, Lemma 6.2].

Further future work could investigate criteria sufficient for a decidable check of cyclic dependency relations for theories of definitions. At the current stage, we suggest to parameterise the cyclicity check by a maximal search depth. Such an algorithm is not complete, as it either finds a cycle, or confirms that there is no cycle, or finds that there may be a cycle of length longer than the maximal search depth.

5 Conclusion

Acyclic definitions entail consistent theories in the Isabelle/HOL theorem prover [KP19, ÅG20]. In this report we show that cycles in type-substitutive dependencies start with a non-type-substitutive step for composable dependency relations. We fix the proof by Kunčar, and discuss why further restrictions, that render acyclicity decidable, do not apply for dependencies by definitions. Ultimately, overloading definitions are no different from other definitions in that they could need a termination argument to aid an automatic checker that cannot decide the acyclicity.

Future work could finalise the formalisation of a cyclicity check of overloading definitions. To the best of our knowledge this would entail the first formally verified proof assistant, by the CakeML tool chain [TMK+19], that supports overloading and has verified model-theoretic conservativity guarantees. The expected result is a verified proof-checker for HOL with ad-hoc overloading similar to the work of Abrahamsson [Abr20], but with overloading definitions. Such a proof checker would permit verified proof-checking of Isabelle/HOL theories, such as those published in the *Archive of Formal Proofs* [EKN+].

Acknowledgements

We thank Andrei Popescu for insightful discussion on Lemma 10. Parts of this work was enabled by the support of the first author by the C.F. Liljewalchs Stipendiestiftelse during his travel to Sydney, Australia.

References

[Abr20] Oskar Abrahamsson. A verified proof checker for higher-order logic. *J. Log. Algebraic Methods Program.*, 112:100530, 2020.

[ÅG20] Johannes Åman Pohjola and Arve Gengelbach. A Mechanised Semantics for HOL with Ad-hoc Overloading. In Elvira Albert and Laura Kovács, editors, *LPAR23. LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 73 of *EPiC Series in Computing*, pages 498–515. EasyChair, 2020.

[BSN$^+$01] Franz Baader, Wayne Snyder, Paliath Narendran, Manfred Schmidt-Schauß, and Klaus U. Schulz. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 445–532. Elsevier and MIT Press, 2001.

[EKN$^+$] Manuel Eberl, Gerwin Klein, Tobias Nipkow, Larry Paulson, and René Thiemann, editors. *Archive of Formal Proofs*. ISSN: 2150-914x.

[GÅW21] Arve Gengelbach, Johannes Åman Pohjola, and Tjark Weber. Mechanisation of Model-theoretic Conservative Extension for HOL with Ad-hoc Overloading. In Claudio Sacerdoti Coen and Alwen Tiu, editors, Proceedings Fifteenth Workshop on *Logical Frameworks and Meta-Languages: Theory and Practice,* Paris, France, 29th June 2020, volume 332 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–17. Open Publishing Association, 2021.

[GW20] Arve Gengelbach and Tjark Weber. Proof-theoretic Conservativity for HOL with Ad-hoc Overloading. In Violet Ka I Pun, Adenilso da Silva Simão, and Volker Stolz, editors, *Theoretical Aspects of Computing - ICTAC 2020 - 17th International Colloquium, December 2 - December 4, 2020, Proceedings*, volume 12545 of *Lecture Notes in Computer Science*, pages 23–42. Springer, 2020.

[KP19] Ondřej Kunčar and Andrei Popescu. A Consistent Foundation for Isabelle/HOL. *J. Autom. Reason.*, 62(4):531–555, 2019.

[Kun15] Ondřej Kunčar. Correctness of Isabelle's Cyclicity Checker: Implementability of Overloading in Proof Assistants. In *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 85–94. ACM, 2015.

[Obu06] Steven Obua. Checking Conservativity of Overloaded Definitions in Higher-Order Logic. In *Term Rewriting and Applications, 17th International Conference, RTA 2006, Seattle, WA, USA, August 12-14, 2006, Proceedings*, pages 212–226. Springer, 2006.

[TMK+19] Yong Kiam Tan, Magnus O. Myreen, Ramana Kumar, Anthony C. J. Fox, Scott Owens, and Michael Norrish. The verified cakeml compiler backend. *J. Funct. Program.*, 29:e2, 2019.

Recent technical reports from the Department of Information Technology

2020-004 Malin Källén, Sverker Holmgren, and Ebba Thora Hvannberg: *Impact of Code Refactoring using Object-Oriented Methodology on a Scientific Computing Application*

2020-003 Sven-Erik Ekström, Carlo Garoni, Adam Jozefiak, and Jesse Perla: *Eigenvalues and Eigenvectors of Tau Matrices with Applications to Markov Processes and Economics*

2020-002 Owe Axelsson, Michal Béres, and Radim Blaheta: *A Boundary Optimal Control Identification Problem*

2020-001 Åsa Cajander, Marta Larusdottir, and Gustaf Hedström: *Digital arbetsmiljö och projektet Nära vård online*

2019-011 Pietro Benedusi, Paola Ferrari, Carlo Garoni, Rolf Krause, and Stefano Serra-Capizzano: *Fast Parallel Solver for the Space-Time IgA-DG Discretization of the Anisotropic Diffusion Equation*

2019-010 Umberto Soverini and Torsten Söderström: *The Frisch Scheme: Time and Frequency Domain Aspects*

2019-009 Kristiina Ausmees: *Evaluation of Methods Handling Missing Data in PCA on Genotype Data: Applications for Ancient DNA*

2019-008 Kristiina Ausmees, Federico Sanchez-Quinto, Mattias Jakobsson, and Carl Nettelblad: *An Empirical Evaluation of Genotype Imputation of Ancient DNA*

2019-007 Malin Källén and Tobias Wrigstad: *Performance of an OO Compute Kernel on the JVM: Revisiting Java as a Language for Scientific Computing Applications (Extended Version)*

2019-006 Umberto Soverini and Torsten Söderström: *Frequency Domain Identification of FIR Models from Noisy Input-Output Data*

2019-005 Giovanni Barbarino, Carlo Garoni, and Stefano Serra-Capizzano: *Block Generalized Locally Toeplitz Sequences: Theory and Applications in the Multidimensional Case*

2019-004 Giovanni Barbarino, Carlo Garoni, and Stefano Serra-Capizzano: *Block Generalized Locally Toeplitz Sequences: Theory and Applications in the Unidimensional Case*

UPPSALA
UNIVERSITET

March 2021
ISSN 1404-3203
http://www.it.uu.se/