Superposition with Datatypes and Codatatypes

Jasmin Christian Blanchette[1,2], Nicolas Peltier[3], and Simon Robillard[4(✉)]

[1] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
[2] Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
[3] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
[4] Chalmers University of Technology, Gothenburg, Sweden
simon.robillard@chalmers.se

Abstract. The absence of a finite axiomatization of the first-order theory of datatypes and codatatypes represents a challenge for automatic theorem provers. We propose two approaches to reason by saturation in this theory: one is a conservative theory extension with a finite number of axioms; the other is an extension of the superposition calculus, in conjunction with axioms. Both techniques are refutationally complete with respect to nonstandard models of datatypes and non-branching codatatypes. They take into account the acyclicity of datatype values and the existence and uniqueness of cyclic codatatype values. We implemented them in the first-order prover Vampire and compare them experimentally.

1 Introduction

The ability to reason about inductive and coinductive datatypes has many applications in program verification, formalization of the metatheory of programming languages, and even formalization of mathematics. Inductive datatypes, or simply *datatypes*, consist of finite values freely generated from constructors. Coinductive datatypes, or *codatatypes*, additionally support infinite values. Non-freely generated (co)datatypes are also useful. All of these variants can be seen as members of a single unifying framework (Section 2).

It is well known that the first-order theory of datatypes cannot be finitely axiomatized. Distinctness, injectivity, and exhaustiveness of constructors are easy to axiomatize, but acyclicity is more subtle, and for induction we would need an axiom schema or a second-order axiom. Codatatypes are also problematic: Besides a coinduction principle that is dual to induction, they are characterized by the existence of all possible infinite values, corresponding intuitively to infinite ground terms. Both datatypes and codatatypes represent a challenge for automatic theorem provers.

Superposition [2] is a highly successful calculus for reasoning about first-order clauses and equality. There has been some work on extending superposition with induction [9, 23], including by Kersani and Peltier [10], and on the axiomatization of datatypes, including by Kovács, Robillard, and Voronkov [11]. In this report, we propose both axiomatizations and extensions of the superposition calculus to support freely and non-freely generated datatypes as well as codatatypes.

We first focus on a conservative extension of the theory with a finite number of first-order axioms that capture the basic properties of constructors, acyclicity of datatype values, uniqueness of cyclic (ω-regular) codatatype values, and existence of all codatatype cyclic values (Section 3). These axioms admit nonstandard models; for example, for the

Peano-style natural numbers freely generated by $\mathsf{zero} : \mathit{nat}$ and $\mathsf{suc} : \mathit{nat} \to \mathit{nat}$, we cannot exclude the familiar nonstandard models of arithmetic, in which arbitrarily many copies of $\mathbb{Z}$ may appear besides $\mathbb{N}$. Similarly, the domains interpreting codatatypes are not guaranteed to contain all infinite acyclic values.

The axiomatization of codatatypes up to a suitable notion of nonstandard models constitutes the first theoretical contribution of this report. Our second, and main, theoretical contribution is an extension of superposition with inference rules to reason about datatypes and codatatypes (Section 4). This is inspired by an acyclicity rule that Robillard presented at the Vampire 2017 workshop [21]. The main distinguishing feature of our rules is that they are (in combination with a few axioms) refutationally complete and their side conditions have some new order restrictions, helping prune the search space. On the other hand, our approach also requires a relaxation of the side conditions of the superposition rule: For clauses of the form $\mathsf{c}(\bar{s}) \approx t \vee \mathcal{C}$, where c is a constructor and the first literal is maximal and positive, superposition inferences onto t must be performed, as in ordered paramodulation [1]. In addition, we propose, for the first time, calculus extensions to reason about codatatypes.

Both the theory extension and the calculus extension are designed to be refutationally complete with respect to nonstandard models of datatypes and *nonbranching* codatatypes—codatatypes whose constructors have at most one corecursive argument (Section 5).

The calculus extension can be integrated into the given clause algorithm that forms the core of a prover's saturation loop (Section 6). The inference partners for the acyclicity and uniqueness rules can be located efficiently. We implemented both the axiomatic and the calculus approaches in the first-order prover Vampire [12] and compare them empirically on Isabelle/HOL [16] benchmarks and on crafted benchmarks (Section 7).

2 Syntax and Semantics

Our setting is a many-sorted first-order logic. We let τ, υ range over simple types (sorts), s, t, u, v range over terms, $\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$ range over function symbols, x, y, z range over variables, and $\mathcal{C}, \mathcal{D}, \mathcal{E}$ range over clauses. Literals are atoms of the form $s \approx t$ or $\neg\, s \approx t$, also written $s \not\approx t$. Clauses are finite disjunctions of literals, viewed as multisets. Substitutions are written in postfix notation, with $s\sigma\theta = (s\sigma)\theta$. The notation $\bar{x}$ represents a tuple $(x_1, \ldots, x_m)$, where $m \geq 0$, and $[m, n]$ denotes the set $\{m, m+1, \ldots, n\}$, where $m \leq n+1$.

A *position p of type τ in t* is a position in t such that $t|_p$ is of type τ. If s, t are terms and P is a set of positions of the same type as s in t, then $t[s]_P$ denotes the term obtained from t by replacing the subterms occurring at a position in P by s: $t[s]_P := s$ if $\varepsilon \in P$; $t[s]_P := t$ if $P = \varnothing$; and $\mathsf{f}(t_1, \ldots, t_n)[s]_P := \mathsf{f}(t_i[s]_{P_i})_{i \in [1,n]}$, with $P_i = \{q \mid i.q \in P\}$ otherwise. Given two positions p and q, we write $p < q$ if p is a proper prefix of q.

Let $\mathcal{C}tr$ be a distinguished set of function symbols, called *constructors*. We reserve the letters $\mathsf{c}, \mathsf{d}, \mathsf{e}$ for constructors. A *constructor position* in t is a position q in t such that for every $p < q$, the head symbol of $t|_p$ is a constructor.

Definition 1. The set of *constructor contexts* of profile $\tau \to \upsilon$ is defined inductively as follows: (1) if t is a term of type υ, then t is a constructor context of profile $\tau \to \upsilon$;

(2) if $\Gamma_1, \ldots, \Gamma_n$ are constructor contexts of profile $\tau \to \tau_i$ and $\mathsf{c} : \tau_1 \times \cdots \times \tau_n \to \upsilon$ is a constructor, then $\mathsf{c}(\Gamma_1, \ldots, \Gamma_n)$ is a constructor context of profile $\tau \to \upsilon$; (3) the hole $\bullet$ is a constructor context of profile $\upsilon \to \upsilon$.

Every constructor context can be written as $\Gamma[\bullet]_P$, where P is a set of constructor positions of the same type in Γ, denoting the positions of $\bullet$ in Γ. It is *empty* if $\varepsilon \in P$, and *constant* if $P = \varnothing$. We write $\Gamma[\bullet]_p$ as an abbreviation for $\Gamma[\bullet]_{\{p\}}$, and we write $\Gamma[t]_P$ to denote the term obtained by replacing every position of P by the term t in the context $\Gamma[\bullet]_P$. Moreover, we write $\tau \rhd \upsilon$ ("υ depends on τ") if there exists a constructor of profile $\tau_1 \times \cdots \times \tau_n \to \upsilon$, with $\tau = \tau_i$ for some $i \in [1,n]$, and $\tau \sim \upsilon$ if $\tau \rhd^* \upsilon$ and $\upsilon \rhd^* \tau$.

Proposition 2. *Let t be a term and let p be a constructor position in t.* $\mathrm{type}(t|_p) \rhd^* \mathrm{type}(t)$. *Consequently, if $\Gamma[\bullet]_P$ is a nonconstant constructor context of profile $\upsilon \to \tau$, then $\upsilon \rhd^* \tau$.*

Proof. The first result is by an immediate induction on p. Then the second result follows from the fact that $P \neq \varnothing$. □

The axioms and rules in this report are parameterized by the following sets. Let $\mathcal{T}_{\mathsf{ind}}$ and $\mathcal{T}_{\mathsf{coind}}$ be disjoint sets of types, intended to model datatypes and codatatypes, respectively, and assume that the codomain of every constructor is in $\mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$. Let $\mathcal{C}tr_{\mathsf{inj}} \subseteq \mathcal{C}tr$ be a set of constructors, denoting injective constructors. Let $\bowtie$ be a binary symmetric and irreflexive relation among constructors; $\mathsf{c} \bowtie \mathsf{d}$ indicates that terms with head symbol c are always distinct from terms with head symbol d. Note that $\bowtie$ is not identical to $\neq$, because the constructors are not necessarily free.

We introduce some properties of interpretations that are intended to capture some of the properties of (co)datatypes. An interpretation $\mathcal{I}$ satisfies

- **Exh** (exhaustiveness) iff, for every type $\tau \in \mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$, $\mathcal{I} \models \bigvee_{i=1}^{m} \exists \bar{x}_i.\ x \approx \mathsf{c}_i(\bar{x}_i)$, where x is a variable of type τ, $\{\mathsf{c}_1, \ldots, \mathsf{c}_m\}$ is the set of constructors of codomain τ, and $\bar{x}_i$ is a (possibly empty) vector of pairwise distinct variables of the appropriate length and types;
- **Inf** (infiniteness) iff, for every type $\tau \in \mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$, the domain of τ is infinite;
- **Acy** (acyclicity, for datatypes) iff, for every type $\tau \in \mathcal{T}_{\mathsf{ind}}$, for every nonempty constructor context $\Gamma[\bullet]_p$ of type τ, where p is a position of type τ in Γ, we have $\mathcal{I} \models \Gamma[x]_p \not\approx x$, where x is a variable of type τ not occurring in Γ;
- **FP** (existence and uniqueness of fixpoints, for codatatypes) iff, for every type $\tau \in \mathcal{T}_{\mathsf{coind}}$, for every nonempty constructor context $\Gamma[\bullet]_P$, $\mathcal{I} \models (\exists x.\, \Gamma[x]_P \approx x) \wedge (\Gamma[x]_P \approx x \wedge \Gamma[y]_P \approx y \Longrightarrow x \approx y)$, where x, y are distinct variables of type τ not occurring in Γ;
- **Dst** (distinctness of constructors) iff, for every pair of constructors c, d of the same codomain such that $\mathsf{c} \bowtie \mathsf{d}$, $\mathcal{I} \models \mathsf{c}(\bar{x}) \not\approx \mathsf{d}(\bar{y})$ where $\bar{x}$ and $\bar{y}$ are disjoint vectors of pairwise distinct variables of the appropriate length and types;
- **Inj** (injectivity) iff, for every n-ary constructor $\mathsf{c} \in \mathcal{C}tr_{\mathsf{inj}}$ and pairwise distinct variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ of the appropriate types, $\mathcal{I} \models \mathsf{c}(x_1, \ldots, x_n) \approx \mathsf{c}(y_1, \ldots, y_n) \Longrightarrow \bigwedge_{i=1}^{n} x_i \approx y_i$.

Most datatypes occurring in practice are recursive, so condition **Inf** is usually satisfied. In particular, it is the case for any freely generated (co)datatype τ with more than one constructor and such that $\tau \rhd^+ \tau$. Conditions **Dst** and **Inj** are defined by finite sets of axioms, but not conditions **Acy** and **FP**. In Section 3, we introduce conservative extensions of the considered formula so that conditions **Acy** and **FP** are satisfied. Then in Section 4, we replace some of these axioms by inference rules.

We assume that $\tau \not\sim \upsilon$ whenever $\tau \in \mathcal{T}_{\mathsf{ind}}$ and $\upsilon \in \mathcal{T}_{\mathsf{coind}}$. Intuitively, this condition means that a datatype cannot be defined by mutual recursion with a codatatype, which is a very natural restriction [7]. If this condition does not hold, it is easy to see that there is no interpretation that satisfies both **Acy** and **FP**. On the other hand, we may have $\tau \rhd^+ \upsilon$ or $\upsilon \rhd^+ \tau$ with $\tau \in \mathcal{T}_{\mathsf{ind}}$ and $\upsilon \in \mathcal{T}_{\mathsf{coind}}$—a datatype can depend on a codatatype or vice versa. There may also exist types not belonging to $\mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$, and the types in $\mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$ may depend on them. Finally, we assume without loss of generality that for each type τ, there exists a ground term t (not necessarily built from constructors) of type τ.

3 Axioms

The axioms Exhaust for exhaustiveness, Dist for distinctness, and Inj for injectivity correspond to the formulas used to express the properties **Exh**, **Dst**, and **Inj** in Section 2. The other axioms are introduced below.

3.1 Acyclicity

For all types τ, υ such that $\tau \sim \upsilon$, we introduce a predicate symbol $\mathsf{sub}_\upsilon^\tau$ on $\tau \times \upsilon$ together with the following axioms, where $\tau \sim \upsilon \sim \upsilon'$ and $\mathsf{c} : \cdots \times \upsilon \times \cdots \to \upsilon'$ is a constructor:

$$\mathsf{Sub}_1\colon\ \mathsf{sub}_\tau^\tau(x,x) \qquad\qquad \mathsf{Sub}_2\colon\ \neg\,\mathsf{sub}_\upsilon^\tau(x,y) \vee \mathsf{sub}_{\upsilon'}^\tau(x,\mathsf{c}(\bar{z},y,\bar{z}'))$$

$$\mathsf{NSub}\colon \neg\,\mathsf{sub}_\tau^{\upsilon'}(\mathsf{c}(\bar{z},x,\bar{z}'),x) \quad \text{if } \tau \in \mathcal{T}_{\mathsf{ind}}$$

Let $\mathsf{Sub} = \mathsf{Sub}_1 \wedge \mathsf{Sub}_2$. The least fixpoint model of Sub is the usual subterm relation for constructor terms. The axiom NSub states that no term of a type in $\mathcal{T}_{\mathsf{ind}}$ may occur at a nonempty constructor position in itself.

Proposition 3. *Let $\Gamma[\bullet]_p$ be a constructor context of type τ, where p is a position of type υ in Γ. Then $\mathsf{Sub} \models \mathsf{sub}_\tau^\upsilon(x,\Gamma[x]_p)$, where x is a variable of type υ.*

Proof. By an immediate induction on Γ. □

Definition 4. An interpretation $\mathcal{I}$ is sub-*minimal* if it satisfies the equivalence $\mathsf{sub}(x,y) \Leftrightarrow \bigvee\{\exists\bar{z}.\, y \approx \Gamma[x]_p \mid \Gamma[\bullet]_p \text{ is a constructor context}\}$, where $\bar{z}$ denotes the vector of variables in Γ that are distinct from x, y.

Proposition 5. *Any* sub-*minimal interpretation satisfies* Sub.

Proof. Let $\mathcal{I}$ be a sub-minimal interpretation. By letting $p = \varepsilon$ in Definition 4, we deduce that $\mathcal{I} \models \mathsf{sub}(x,x)$. Furthermore, for any valuation η such that $\mathcal{I},\eta \models \mathsf{sub}(x,y)$, we have $\mathcal{I},\eta \models \exists \bar{z}''.\ y \approx \Gamma[x]_p$, for some constructor context $\Gamma[\bullet]_p$, thus $\mathcal{I},\eta' \models y \approx \Gamma[x]_p$, for some extension η' of η (we assume by renaming that $\bar{z}''$ is disjoint from $\bar{z}$ and $\bar{z}'$). Consequently $\mathcal{I},\eta' \models \mathsf{c}(\bar{z},y,\bar{z}') \approx \mathsf{c}(\bar{z},\Gamma[x]_p,\bar{z}')$, hence $\mathcal{I},\eta' \models \mathsf{c}(\bar{z},y,\bar{z}') \approx \mathsf{c}(\bar{z},\Gamma,\bar{z}')[x]_{i.p}$, with $i = |\bar{z}|+1$. It is clear that $\mathsf{c}(\bar{z},\Gamma,\bar{z}')[\bullet]_{i.p}$ is a constructor context, hence by Definition 4, we have $\mathcal{I},\eta' \models \mathsf{sub}(x,\mathsf{c}(\bar{z},\Gamma,\bar{z}')[x]_{i.p})$, thus $\mathcal{I},\eta' \models \mathsf{sub}(x,\mathsf{c}(\bar{z},y,\bar{z}'))$, i.e., $\mathcal{I},\eta \models \mathsf{sub}(x,\mathsf{c}(\bar{z},y,\bar{z}'))$, since $\bar{z}''$ is disjoint from $\bar{z}$ and $\bar{z}'$. Consequently, $\mathcal{I} \models \mathsf{Sub}$. □

3.2 Contexts and Fixpoints

For every pair of types $\tau,\upsilon \in \mathcal{T}_{\mathsf{coind}}$ with $\tau \sim \upsilon$, we introduce a type $\boxed{\tau}_\upsilon$ to denote contexts $\Gamma[\bullet]_P$ of profile $\upsilon \to \tau$. Let hole_υ (for every $\upsilon \in \mathcal{T}_{\mathsf{coind}}$) be a constant of type $\boxed{\upsilon}_\upsilon$, denoting an empty context. All constructors $\mathsf{c} : \tau_1 \times \cdots \times \tau_n \to \tau$ and types υ such that $\exists i\, \upsilon \rhd^* \tau_i$ are associated with new n-ary constructors $\boxed{\mathsf{c}}_\upsilon : \upsilon_1 \times \cdots \times \upsilon_n \to \boxed{\tau}_\upsilon$, where for every $i \in [1,n]$, $\upsilon_i = \boxed{\tau_i}_\upsilon$ if $\upsilon \rhd^* \tau_i$ and $\upsilon_i = \tau_i$ otherwise. Let $\mathsf{app}^\tau_\upsilon : \boxed{\tau}_\upsilon \times \upsilon \to \tau$, $\mathsf{cyc}_\upsilon : \boxed{\upsilon}_\upsilon \to \upsilon$, and $\mathsf{cst}^\tau_\upsilon : \tau \to \boxed{\tau}_\upsilon$ be new function symbols. As usual, type indices are often omitted for readability. Intuitively, if y denotes the context $\Gamma[\bullet]_P$, then $\mathsf{app}(x,y)$ denotes the term $\Gamma[x]_P$, $\mathsf{cyc}(y)$ denotes the fixpoint of $\Gamma[\bullet]_P$, and $\mathsf{cst}^\tau_\upsilon$ denotes a constant context (i.e., a context $\Gamma[\bullet]_P$ with $P = \varnothing$).

We consider the following axioms, where $\upsilon \in \mathcal{T}_{\mathsf{coind}}$ and x, y, x_i, z_i are pairwise distinct variables of the appropriate types:

$$\mathsf{App}_1\colon\ \mathsf{app}^\tau_\upsilon(\mathsf{cst}^\tau_\upsilon(x),y) \approx x \qquad \mathsf{App}_2\colon\ \mathsf{app}^\upsilon_\upsilon(\mathsf{hole}_\upsilon,y) \approx y$$

$$\mathsf{App}_3\colon\ \mathsf{app}^\tau_\upsilon(\boxed{\mathsf{c}}_\upsilon(x_1,\ldots,x_n),y) \approx \mathsf{c}(t_1,\ldots,t_n)$$
$$\text{if } \mathsf{c} : \tau_1 \times \cdots \times \tau_n \to \tau \text{ is a constructor and } \exists i\, \upsilon \rhd^* \tau_i$$
$$\text{with } t_i = \mathsf{app}^{\tau_i}_\upsilon(x_i,y) \text{ if } \upsilon \rhd^* \tau_i \text{ and } t_i = x_i \text{ otherwise}$$

$$\mathsf{Uniq}\colon\ x \approx \mathsf{hole}_\upsilon \vee y \not\approx \mathsf{app}^\upsilon_\upsilon(x,y) \vee z \not\approx \mathsf{app}^\upsilon_\upsilon(x,z) \vee y \approx z$$

$$\mathsf{Cycl}\colon\ \mathsf{cyc}_\upsilon(x) \approx \mathsf{app}^\upsilon_\upsilon(x,\mathsf{cyc}_\upsilon(x))$$

$$\mathsf{Hole}_1\colon\ \mathsf{hole}_\upsilon \not\approx \mathsf{cst}^\upsilon_\upsilon(x) \qquad \mathsf{Hole}_2\colon\ \mathsf{hole}_\upsilon \not\approx \boxed{\mathsf{c}}_\upsilon(x_1,\ldots,x_n) \quad \text{if } \mathsf{c} : \cdots \to \upsilon$$

Let $\mathsf{App} = \mathsf{App}_1 \wedge \mathsf{App}_2 \wedge \mathsf{App}_3$ and $\mathsf{Hole} = \mathsf{Hole}_1 \wedge \mathsf{Hole}_2$.

Example 6. Let $\mathsf{c} : \tau_0 \times \upsilon \to \tau$ be a constructor, with $\upsilon \rhd^* \tau_0$. Then the profile of $\boxed{\mathsf{c}}_\upsilon$ is $\boxed{\tau_0}_\upsilon \times \boxed{\upsilon}_\upsilon \to \boxed{\tau}_\upsilon$. The term $t := \boxed{\mathsf{c}}_\upsilon(\mathsf{cst}^{\tau_0}_\upsilon(x),\mathsf{hole}_\upsilon)$ encodes the constructor context $\mathsf{c}(x,\bullet)$. If $\mathsf{a} : \upsilon$, then

$$\begin{aligned}\mathsf{app}^\tau_\upsilon(t,\mathsf{a}) &=_{\mathsf{App}} \mathsf{c}(\mathsf{app}^{\tau_0}_\upsilon(\mathsf{cst}^{\tau_0}_\upsilon(x),\mathsf{a}),\mathsf{app}^\upsilon_\upsilon(\mathsf{hole}_\upsilon,\mathsf{a}))\\ &=_{\mathsf{App}} \mathsf{c}(x,\mathsf{app}^\upsilon_\upsilon(\mathsf{hole}_\upsilon,\mathsf{a}))\\ &=_{\mathsf{App}} \mathsf{c}(x,\mathsf{a})\end{aligned}$$

where $=_{\mathsf{App}}$ denotes equality modulo App (i.e., $s =_{\mathsf{App}} t \Leftrightarrow \mathsf{App} \models s \approx t$). By contrast, if $\upsilon \not\rhd^* \tau_0$, the profile of $\boxed{\mathsf{c}}_\upsilon$ is $\tau_0 \times \boxed{\upsilon}_\upsilon \to \boxed{\tau}_\upsilon$, and the above context is encoded by $t' := \boxed{\mathsf{c}}_\upsilon(x,\mathsf{hole}_\upsilon)$, with $\mathsf{app}^\tau_\upsilon(t',\mathsf{a}) =_{\mathsf{App}} \mathsf{c}(x,a)$. The difference between the two cases is that if $\upsilon \not\rhd^* \tau_0$, then all the contexts of profile $\upsilon \to \tau_0$ are constant. Thus they may be replaced by terms of type τ_0. There is no need to encode them using the function cst.

Remark 7. The axiom Uniq can be replaced by

$$\mathsf{Uniq}'\colon\ x \approx \mathsf{hole}_\upsilon \vee y \not\approx \mathsf{app}^\upsilon_\upsilon(x,y) \vee y \approx \mathsf{cyc}_\upsilon(x)$$

for $\upsilon \in \mathcal{T}_{\mathsf{coind}}$. Indeed, it is clear that $\mathsf{Uniq} \wedge \mathsf{Cycl} \Leftrightarrow \mathsf{Uniq}' \wedge \mathsf{Cycl}$.

Proposition 8. *Let t be a ground term of type $\boxed{\tau}_\upsilon$, with $\tau,\upsilon \in \mathcal{T}_{\mathsf{coind}}$ and $\tau \sim \upsilon$. There exists a ground constructor context $\Gamma[\bullet]_P$ such that $\mathsf{App} \models \mathsf{app}^\tau_\upsilon(t,x) \approx \Gamma[x]_P$, for all variables $x:\upsilon$. Furthermore, if $t \neq \mathsf{hole}_\tau$, then $\varepsilon \notin P$.*

Proof. The proof is by induction on t. If $t = \mathsf{hole}_\tau$, then $\mathsf{App}_2 \models \mathsf{app}(t,x) \approx x$. Let $\Gamma[\bullet]_P = \bullet$ with $P = \{\varepsilon\}$. By definition, $\Gamma[x]_P = x$ hence $\mathsf{App} \models \mathsf{app}(t,x) \approx \Gamma[x]_P$. If $t = \mathsf{cst}^\tau_\upsilon(s)$, then $\mathsf{App}_1 \models \mathsf{app}^\tau_\upsilon(t,x) \approx s$. Let $\Gamma = s$ and $P = \varnothing$. By definition, $\Gamma[x]_P = u = s$, hence $\mathsf{App} \models \mathsf{app}(t,x) \approx \Gamma[x]_P$. Now assume that $t = \boxed{\mathsf{c}}(t_1,\ldots,t_n)$. We have $\mathsf{App}_3 \models \mathsf{app}(t,x) \approx \mathsf{c}(t'_1,\ldots,t'_n)$ with $t'_i = \mathsf{app}(t_i,x)$ if $\upsilon \rhd^* \tau_i$ and $t'_i = t_i$ otherwise. Let $i \in [1,n]$ such that $\upsilon \rhd^* \tau_i$. Then we have necessarily $\tau_i \in \mathcal{T}_{\mathsf{coind}}$, hence by the induction hypothesis, there exists a ground constructor context $\Gamma_i[\bullet]_{P_i}$ such that $\mathsf{App} \models \mathsf{app}(t_i,x) \approx \Gamma_i[x]_{P_i}$. If $\upsilon \not\rhd^* \tau_i$, then we let $\Gamma_i = t_i$ and $P_i = \varnothing$. Let $\Gamma = \mathsf{c}(\Gamma_1,\ldots,\Gamma_n)$ and $P = \{i.q \mid i \in [1,n], q \in P_i\}$. It is clear that $\Gamma[x]_P \approx \mathsf{c}(\Gamma_i[x]_{P_i})_{i\in[1,n]}$, thus $\mathsf{App} \models \mathsf{app}(t,x) \approx \Gamma[x]_P$. It is easy to check that the second part of the proposition is satisfied in every case. □

Proposition 9. *Let $\Gamma[\bullet]_P$ be a constructor context of profile $\upsilon \to \tau$. If $\tau,\upsilon \in \mathcal{T}_{\mathsf{coind}}$ and $\tau \sim \upsilon$, there exists a term $u : \boxed{\tau}_\upsilon$ such that $\mathsf{App} \models \mathsf{app}(u,x) \approx \Gamma[x]_P$, for all variables $x:\upsilon$. Furthermore, if $\varepsilon \notin P$, the head symbol of u is either* cst *or a symbol* $\boxed{\mathsf{c}}$.

Proof. The proof is by induction on Γ. If $\varepsilon \in P$, then $\Gamma[x]_P = x$ and $\tau = \upsilon$. Let $u = \mathsf{hole}_\tau$. By definition, we have $\mathsf{App}_2 \models \mathsf{app}(u,x) \approx x$. If $P = \varnothing$, then $\Gamma[x]_P = \Gamma$ and Γ is a term of type τ. Let $u = \mathsf{cst}^\tau_\upsilon(\Gamma)$. We have $\mathsf{App}_1 \models \mathsf{app}^\tau_\upsilon(u,x) \approx \Gamma$. Otherwise, P contains a nonempty position, hence Γ must be of the form $\mathsf{c}(\Gamma_1,\ldots,\Gamma_n)$ for some constructor $\mathsf{c} : \tau_1 \times \cdots \times \tau_n \to \tau$. Furthermore, there exists $i \in [1,n]$ such that $\upsilon \rhd^* \tau_i$. Let $P_i = \{q \mid i.q \in P\}$, for $i \in [1,n]$. Let $i \in [1,n]$ such that $\tau_i \not\rhd^* \upsilon$. We have $\tau_i \in \mathcal{T}_{\mathsf{coind}}$, hence, by the induction hypothesis, there exists a term u_i such that $\mathsf{App} \models \mathsf{app}(u_i,x) \approx \Gamma_i[x]_{P_i}$. If $\tau_i \not\rhd^* \upsilon$, then all the constructor contexts of profile $\upsilon \to \tau_i$ are constant, thus Γ_i is a term, and we let $u_i = \Gamma_i$. Let $u = \boxed{\mathsf{c}}(u_1,\ldots,u_n)$. By App_3, we deduce $\mathsf{App} \models \mathsf{app}(\boxed{\mathsf{c}}(u_1,\ldots,u_n),x) \approx \mathsf{c}(u_1,\ldots,u_n)$, where $u_i = \Gamma_i[x]_{P_i}$ if $\tau_i \sim \upsilon$ and $u_i = \Gamma_i$ otherwise. If P contains a position of the form $i.q$, then necessarily $\tau_i \sim \upsilon$, thus we have $\Gamma[x]_P = \mathsf{c}(u_1,\ldots,u_n)$. Hence $\mathsf{App} \models \mathsf{app}(u,x) \approx \Gamma[x]_P$. It is clear that the second part of the proposition holds in every case. □

3.3 Soundness and Completeness

We prove that the above axioms indeed capture all the intended properties.

Lemma 10 (Soundness of the Axioms). *If interpretation $\mathcal{I}$ satisfies* **Acy** *and* **FP**, *there exists a* sub*-minimal extension $\mathcal{J}$ of $\mathcal{I}$ validating* Sub, NSub, App, Uniq, Cycl, *and* Hole.

Proof. To simplify notations, we assume that $\mathcal{I}$ is a term model (on an extended signature, with an infinite set of new constant symbols denoting elements of the domain). We define the extension $\mathcal{J}$ of $\mathcal{I}$ and check that it fulfills all the desired properties:

- The interpretation of sub is defined in such a way that $\mathcal{J}$ is sub-minimal. By Proposition 5, $\mathcal{J} \models \mathsf{Sub}$. If $\mathcal{J} \not\models \mathsf{NSub}$, then $\mathcal{J} \models \exists \bar{z}, x, \bar{z}'.\ \mathsf{sub}(\mathsf{c}(\bar{z}, x, \bar{z}'), x)$. By definition of the interpretation of sub, this entails that $\mathcal{I} \models \exists \bar{z}, x, \bar{z}', \bar{y}.\ x \approx \Gamma[\mathsf{c}(\bar{z}, x, \bar{z}')]_p$, for some constructor context Γ and position p, where $\bar{y}$ denotes the vector of variables in Γ. This contradicts the fact that $\mathcal{I}$ satisfies **Acy**.
- The domain of $\boxed{\tau}_\upsilon$ is the set of ground terms of type $\boxed{\tau}_\upsilon$ defined on the signature, with $\mathsf{f}^{\mathcal{J}}(\bar{t}) = \mathsf{f}(\bar{t})$, for any function symbol f of codomain $\boxed{\tau}_\upsilon$ and for any vector of ground terms $\bar{t}$. This set is not empty since it contains $\mathsf{cst}^\tau_\upsilon(t)$ for every ground term $t : \tau$ (and all types are inhabited). Furthermore, $\mathcal{J} \models \mathsf{Hole}$, since two ground terms with distinct heads are necessarily (syntactically) distinct.
- We define the interpretation of $\mathsf{app}(u, v)$ by induction on u, using the equations in App as rewrite rules, from the left to the right, replacing the constructors c in the right-hand side by their interpretation in $\mathcal{I}$. It is clear that app is unambiguously and completely defined, and by definition $\mathcal{J} \models \mathsf{App}$. We check that $\mathcal{J} \models \mathsf{Uniq}$. Let u be a ground term of type $\boxed{\tau}_\upsilon$ distinct from hole_υ. By Proposition 8, there exists a ground constructor context $\Gamma[\bullet]_P$ such that $\mathsf{App} \models \mathsf{app}(u, x) \approx \Gamma[x]_P$, with $\varepsilon \notin P$. Thus $\mathcal{J} \models \mathsf{app}(u, y) \approx y \wedge \mathsf{app}(u, z) \approx z \Longrightarrow \Gamma[y]_P \approx y \wedge \Gamma[z]_P \approx z$, where y, z are variables some type $\tau \in \mathcal{T}_{\mathsf{coind}}$. Since $\mathcal{I}$ satisfies condition **FP**, this entails that $\mathcal{J} \models \mathsf{app}(u, y) \approx y \wedge \mathsf{app}(u, z) \approx z \Longrightarrow y \approx z$. Thus $\mathcal{J} \models \mathsf{Uniq}$.
- Let u be a ground term of type $\boxed{\tau}_\upsilon$ with $\tau \in \mathcal{T}_{\mathsf{coind}}$. By Proposition 8, there exist a ground constructor context $\Gamma[\bullet]_P$ such that $\mathsf{App} \models \mathsf{app}(u, v) \approx \Gamma[v]_P$. We define the interpretation of $\mathsf{cyc}_\tau(u)$ as the unique fixpoint of $\Gamma[\bullet]_P$. By definition, we have $\mathcal{J} \models \Gamma[\mathsf{cyc}_\tau(u)]_P \approx \mathsf{cyc}_\tau(u)$, therefore $\mathcal{J} \models \mathsf{app}(u, \mathsf{cyc}_\tau(u)) \approx \mathsf{cyc}_\tau(u)$. Hence $\mathcal{J} \models \mathsf{Cycl}$.

□

Lemma 11 (Completeness of the Axioms). *Any model of the set of axioms* $\{\mathsf{Sub}, \mathsf{NSub}, \mathsf{App}, \mathsf{Uniq}, \mathsf{Cycl}, \mathsf{Hole}\}$ *fulfills* **Acy** *and* **FP**.

Proof. Let $\mathcal{I}$ be a model of $\{\mathsf{Sub}, \mathsf{NSub}, \mathsf{App}, \mathsf{Uniq}, \mathsf{Cycl}, \mathsf{Hole}\}$.

Acyclicity: If $\mathcal{I}$ does not satisfy condition **Acy**, there exists a nonempty constructor context $\Gamma[\bullet]_p$ of type $\tau \in \mathcal{T}_{\mathsf{ind}}$ such that $\mathcal{I} \models \exists x, \bar{z}.\ x \approx \Gamma[x]_p$, where $\bar{z}$ denotes the vector of variables in Γ. Since $p \neq \varepsilon$, p is of the form $q.i$, for some number i, and the subcontext at position q in $\Gamma[\bullet]_P$ is of the form $\mathsf{c}(\bar{u}, \bullet, \bar{v})$, for some constructor c. By definition $\Gamma[\mathsf{c}(\bar{u}, x, \bar{v})]_q = \Gamma[x]_p$, thus $\mathcal{I} \models \exists x, \bar{z}.\ x \approx \Gamma[\mathsf{c}(\bar{u}, x, \bar{v})]_q$. By Proposition 3, we deduce that $\mathcal{I} \models \mathsf{sub}(\mathsf{c}(\bar{u}, x, \bar{v}), x)$, yielding a contradiction with the axiom NSub.

Let $\Gamma[\bullet]_P$ be a nonempty constructor context of profile $\tau \to \tau$. By Proposition 9, there exists a term $t' : \boxed{\tau}_\tau$ such that $\mathsf{App} \models \mathsf{app}(t', x) \approx \Gamma[x]_P$, and we have $\mathcal{I} \models \mathsf{app}(t', x) \approx \Gamma[x]_P$ $(*)$. Note that, by Hole, $\mathcal{I} \models t' \not\approx \mathsf{hole}_\tau$, since the head symbol of t' is cst or $\boxed{\mathsf{c}}$.

- **Existence of fixpoint:** By $(*)$, $\mathcal{I} \models \mathsf{app}(t', \mathsf{cyc}(t')) \approx \Gamma[\mathsf{cyc}(t')]_P$. By Cycl, we deduce that $\mathcal{I} \models \mathsf{cyc}(t') \approx \Gamma[\mathsf{cyc}(t')]_P$, thus $\mathcal{I} \models \exists x.\ \Gamma[x]_P = x$.

- **Uniqueness:** By $(*)$, $\mathcal{I} \models \Gamma[x]_P \approx x \wedge \Gamma[y]_P \approx y \Longrightarrow \mathsf{app}(t', x) \approx x \wedge \mathsf{app}(t', y) \approx y$. By Uniq, we deduce that $\mathcal{I} \models \Gamma[x]_P \approx x \wedge \Gamma[y]_P \approx y \Longrightarrow x \approx y$.

□

Lemma 12 (Completeness of the Theory). *Let $\mathcal{T}$ be the theory of free constructors, as defined by the properties* **Exh***,* **Inf***,* **Acy***,* **FP***,* **Dst***, and* **Inj***, with $\mathcal{C}tr_{\mathsf{inj}} = \mathcal{C}tr$ and $\mathsf{c} \bowtie \mathsf{d}$ for all distinct constructors c and d. If S is a first-order sentence in which the only symbols occurring are constructors and equality ($\approx$), then either $\mathcal{T} \models S$ or $\mathcal{T} \models \neg S$.*

Comon and Lescanne [8] provide a decision procedure for equational formulas over finite and infinite trees, which correspond respectively to freely generated datatypes and codatatypes. It is based on a collection of equivalence-preserving transformation rules for eliminating quantifiers and normalizing the formulas. The set of formulas $T = \{\mathsf{Dist}, \mathsf{Inj}, \mathsf{Exhaust}, \mathsf{Sub}, \mathsf{NSub}, \mathsf{App}, \mathsf{Uniq}, \mathsf{Cycl}, \mathsf{Hole}\}$ forms the axiomatization of a conservative extension of the theory of (co)datatypes. We can thus derive a decision procedure for testing satisfiability of first-order sentences S containing only constructors symbols and the equality predicate in the above theory. By interleaving the steps of two fair saturation procedures of the superposition calculus, the first over $S \cup T$ and the second over $\neg S \cup T$, one of the two attempts is guaranteed to derive a refutation in finite time.

4 Inference Rules

As an alternative to the above axiomatization, we propose an extension of the superposition calculus [2] with dedicated rules. Unless otherwise noted, the usual conventions of superposition apply. The standard notion of redundancy is used, with respect to the theory of equality. The notation $s \not\approx t$ indicates that the literal is maximal in its clause, after the substitution σ has been applied, and nothing is selected, or it is selected, whereas $s \approx t$ indicates that the literal is strictly maximal in its clause, after σ, and not selected. We let $[\neg]\, s \approx t$ stand for either $s \approx t$ or $s \not\approx t$.

4.1 Superposition

We denote by $\mathcal{SP}$ the usual rules of the superposition calculus, called Sup, EqRes, and EqFact below, with a slight relaxation of the application conditions of Sup: Superposition inside the nonmaximal term of an equation is allowed if the head symbol of the maximal term is a constructor. This ensures that in the rewrite system built from saturated clause sets for defining a model, the right-hand side of every rule is irreducible if the head of the left-hand side is a constructor. This property is crucial for the completeness results.

Thus, our superposition rule is as follows:

$$\frac{u \approx v \vee \mathcal{D} \qquad [\neg]\, s[u'] \approx t \vee \mathcal{C}}{\sigma([\neg]\, s[v] \approx t \vee \mathcal{D} \vee \mathcal{C})}\ \mathsf{Sup}$$

where $\sigma = \mathrm{mgu}\{u \stackrel{?}{=} u'\}$, u' is not a variable, and $\sigma(u) \not\preceq \sigma(v)$; moreover, $\sigma(s[u']) \not\preceq \sigma(t)$ if $[\neg]$ is $\neg$ or if the head symbol of t is not a constructor. The equality resolution and equality factoring rules are the standard ones.

The equality resolution rule is as usual:

$$\frac{s \not\approx s' \vee \mathcal{C}}{\sigma(\mathcal{C})}\ \mathsf{EqRes}$$

where $\sigma = \mathrm{mgu}\,\{s, s'\}$. Similarly for the equality factoring rule:

$$\frac{s \approx t \vee s' \approx u \vee \mathcal{C}}{\sigma(t \not\approx u \vee s \approx u \vee \mathcal{C})}\ \mathsf{EqFact}$$

where $\sigma = \mathrm{mgu}\,\{s, s'\}$ and $\sigma(s) \not\preccurlyeq \sigma(t)$. Exceptionally, $s \approx t$ is required only to be maximal in the premise, not strictly maximal.

4.2 Infiniteness

The next rule captures infiniteness of (co)datatypes:

$$\frac{\left(\bigvee_{i=1}^{n} x \approx t_i\right) \vee \mathcal{C}}{\mathcal{C}}\ \mathsf{Inf}$$

if x is a variable of a type $\tau \in \mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$ and does not occur in $\mathcal{C}$ or $t_1, \ldots, t_n$.

Lemma 13 (Soundness of Inf). *Let N be a clause set, and let $\mathcal{I}$ be a model of N satisfying* **Inf**. *If $\mathcal{C}$ is derived from N by* Inf, *then $\mathcal{I} \models \mathcal{C}$.*

Proof. Since $\mathcal{I}$ satisfies **Inf**, the domain of τ is infinite. Therefore, for every valuation η, $\mathcal{I}, \eta \not\models \forall x \bigvee_{i=1}^{n} x \approx t_i$ (since x does not occur in $t_1, \ldots, t_n$), hence $\mathcal{I}, \eta \models \mathcal{C}$ (since x does not occur in $\mathcal{C}$). □

4.3 Distinctness

The distinctness property of constructors takes the form of a unary and a binary rule:

$$\frac{\mathsf{c}(\bar{s}) \approx t \vee \mathcal{C}}{\mathcal{C}\sigma}\ \mathsf{Dist}_1$$

if $\sigma = \mathrm{mgu}\,\{t \stackrel{?}{=} \mathsf{d}(\bar{x})\}$, where $\mathsf{c} \bowtie \mathsf{d}$ and $\bar{x}$ is a vector of fresh pairwise distinct variables; and

$$\frac{\mathsf{d}(\bar{t}) \approx u' \vee \mathcal{D} \quad \mathsf{c}(\bar{s}) \approx u \vee \mathcal{C}}{(\mathcal{C} \vee \mathcal{D})\sigma}\ \mathsf{Dist}_2$$

if $\mathsf{c} \bowtie \mathsf{d}$, $\sigma = \mathrm{mgu}\,\{u \stackrel{?}{=} u'\}$, $\mathsf{c}(\bar{s})\sigma \not\preccurlyeq u\sigma$, and $\mathsf{d}(\bar{t})\sigma \not\preccurlyeq u'\sigma$.

Proposition 14 (Soundness of Dist_1 and Dist_2). *Let N be a clause set, and let $\mathcal{I}$ be a model of N satisfying* **Dst**. *If a clause $\mathcal{C}$ is derived from N by* Dist_1 *or* Dist_2, *then $\mathcal{I} \models \mathcal{C}$.*

Proof. It is easy to check that the conclusion can be derived by superposition and equality resolution from the premises and the axiom Dist. □

Remark 15. If t is not a variable, the premise of Dist_1 is redundant after the rule is applied. Unifying t with $\mathsf{c}(\bar{x})$ can be useful when t is a variable. For example, from the clause $\mathsf{c}(x) \approx x$, we can derive $\square$ by unifying x with $\mathsf{d}(\bar{y})$, where $\mathsf{d} \bowtie \mathsf{c}$.

4.4 Injectivity

The injectivity property of constructors is also captured by two rules:

$$\frac{\mathsf{c}(s_1,\ldots,s_m) \approx t \vee \mathcal{C}}{(s_i \approx x_i \vee \mathcal{C})\sigma}\ \mathsf{Inj}_1$$

if $i \in [1,m]$, $\mathsf{c} \in \mathcal{C}tr_{\mathsf{inj}}$, $\sigma = \mathrm{mgu}\,\{t \stackrel{?}{=} \mathsf{c}(x_1,\ldots,x_m)\}$, and $x_1,\ldots,x_m$ are fresh pairwise distinct variables; and

$$\frac{\mathsf{c}(s_1,\ldots,s_m) \approx u' \vee \mathcal{D} \quad \mathsf{c}(t_1,\ldots,t_m) \approx u \vee \mathcal{C}}{(s_i \approx t_i \vee \mathcal{C} \vee \mathcal{D})\sigma}\ \mathsf{Inj}_2$$

if $i \in [1,m]$, $\mathsf{c} \in \mathcal{C}tr_{\mathsf{inj}}$, $\sigma = \mathrm{mgu}\,\{u \stackrel{?}{=} u'\}$, $u\sigma \not\approx \mathsf{c}(\bar{s})\sigma$, and $u'\sigma \not\approx \mathsf{c}(\bar{t})\sigma$.

Proposition 16 (Soundness of Inj_1 and Inj_2). *Let N be a clause set, and let $\mathcal{I}$ be a model of N satisfying* **Inj**. *If a clause $\mathcal{C}$ is derived from N by Inj_1 or Inj_2, then $\mathcal{I} \models \mathcal{C}$.*

Proof. It is easy to check that the conclusion can be derived by superposition and equality resolution from the premises and the axiom Inj. □

Remark 17. If Inj_1 is applied on every argument $i \in [1,m]$ and t is not a variable, the premise becomes redundant and can be removed. Unifying t with the term $\mathsf{c}(x_1,\ldots,x_m)$ is useful when t is a variable. For example, given the clause $\mathsf{c}(x,\mathsf{a}) \approx x$, we can derive $\mathsf{a} \approx x_2$ by Inj_1, from which $\square$ can be derived by Inf.

4.5 Acyclicity

The acyclicity rule attempts to detect constraints that would force a datatype value to be cyclic. The simplest example is a clause of the form $\Gamma[s] \approx s$, where Γ is a nonempty constructor context. More generally, the clauses

$$s_1 \approx \Gamma_1[s_2] \qquad s_2 \approx \Gamma_2[s_3] \qquad \cdots \qquad s_{n-1} \approx \Gamma_{n-1}[s_n] \qquad s_n \approx \Gamma_n[s_1]$$

entail a constraint $s_1 \approx \Gamma_1[\Gamma_2[\cdots[\Gamma_{n-1}[\Gamma_n[s_1]]]\cdots]]$. Moreover, the rule must support variables and nonunit clauses, and it should be finitely branching if we want to incorporate it in saturation-based provers—i.e., the set of clauses derivable from a given finite set of premises by a single rule should be finite. Finally, clauses of the form $\Gamma[x] \approx s \vee \mathcal{C}$, where x occurs in $\mathcal{C}$, are problematic, because there are infinitely many instantiations of x that can result in a cyclic constraint: s, $\mathsf{c}(s)$, $\mathsf{c}(\mathsf{c}(s))$, etc. To cope with all these subtleties, we first need to develop a considerable theoretical apparatus before we can even state the rule.

Definition 18. A *chain* built on a nonempty sequence of (variable-disjoint) clauses $(\mathcal{C}_1,\ldots,\mathcal{C}_n)$ under condition $\mathcal{D}$ is a sequence $(t_1,\ldots,t_{n+1})$ of terms satisfying the following conditions:

1. for every $i \in [1,n]$, $\mathcal{C}_i$ is of the form $s_i \approx \Gamma_i[s'_{i+1}]_{p_i} \vee \mathcal{C}'_i$, where p_i is a nonempty constructor position in Γ_i;

2. there exists a substitution σ such that either (a) σ is an mgu of $E = \{s'_i \stackrel{?}{=} s_i \mid i \in [2,n]\}$ or (b) σ is an mgu of $\{s'_{n+1} \stackrel{?}{=} s_1\} \cup E$;
3. $t_i = s_i\sigma$ for $i \in [1,n]$ and $t_{n+1} = s'_{n+1}\sigma$;
4. $\mathcal{D} = \bigvee_{i=1}^{n} \mathcal{C}'_i\sigma$;
5. $\text{type}(t_1) \sim \cdots \sim \text{type}(t_{n+1})$;
6. $(\Gamma_i[s'_{i+1}]_{p_i} \approx s_i)\sigma$ is maximal in $\mathcal{C}_i\sigma$, and no negative literal is selected in $\mathcal{C}_i\sigma$;
7. $s_i\sigma \not\succ \Gamma_i[s'_{i+1}]_{p_1}\sigma$, for $i \in [1,n]$;
8. for every $i \in [2,n]$, s'_i is not a variable.

The expression $\Gamma_1[\cdots[\Gamma_n[\bullet]_{p_n}]\cdots]_{p_1}\sigma$ is the chain's constructor context, σ is its mgu, and $p_1.\cdots.p_n$ is its constructor position. If $t_1 = t_{n+1}$, the sequence is called a *cycle*. A chain is *direct* if $t_i \neq t_j$ for all $i, j \in [1,n+1]$ with $i \neq j$ and $\{i,j\} \neq \{1,n+1\}$, and *variable-ended* if s'_{n+1} is a variable.

Remark 19. Conditions 5 to 8 are optional. They help prune the search space.

Remark 20. It is tempting to replace the condition $s_i\sigma \not\succ \Gamma_i[s'_{i+1}]_{p_1}\sigma$ for $i > 1$ by the stronger condition $\Gamma_i[s'_{i+1}]_{p_1}\sigma \succ s_i\sigma$, because if the latter condition does not hold, the Superposition rule applies into s'_i generating a clause $(\Gamma_{i-1}[\Gamma_i[s'_{i+1}]_{p_i}]_{p_{i-1}} \approx s_{i-1} \vee \mathcal{C}'_{i-1} \vee \mathcal{C}'_i)\theta$, where σ is an instance of θ, and we could construct a smaller chain with the same first and last terms without using $\mathcal{C}_i$. But this is not compatible with the redundancy criteria. For example, given $\{\mathsf{f}(x,1) \approx \mathsf{c}(\mathsf{g}(x)), \mathsf{g}(x) \approx \mathsf{c}(\mathsf{f}(x,x)), \mathsf{f}(0,1) \approx \mathsf{c}(\mathsf{c}(\mathsf{f}(0,0)))\}$ with $\mathsf{g}(0) \succ \mathsf{c}(\mathsf{f}(0,0))$, $\mathsf{c}(\mathsf{f}(1,1)) \succ \mathsf{g}(1)$, no chain from $\mathsf{f}(1,1)$ to $\mathsf{f}(1,1)$ could be derived (with the strengthened condition), because $\mathsf{f}(x,1) \approx \mathsf{c}(\mathsf{c}(\mathsf{f}(x,x)))$ is redundant (assuming we have $x \approx 0 \vee x \approx 1$). Indeed, $\mathsf{f}(0,1) \approx \mathsf{c}(\mathsf{c}(\mathsf{f}(0,0)))$ occurs in the set, and $\mathsf{f}(1,1) \approx \mathsf{c}(\mathsf{c}(\mathsf{f}(1,1)))$ can easily be derived from smaller instances by substitutivity.

We state some basic properties of chains:

Proposition 21. *Let $(t_1,\ldots,t_{n+1})$ be a chain, and let $i, j \in [1,n+1]$, with $i \leq j$. With the notations of Definition 18, we have*

$$\mathcal{C}_1,\ldots,\mathcal{C}_n \models \mathcal{D} \vee (t_i \approx \Gamma_i[\cdots[\Gamma_{j-1}[t_j]_{p_{j-1}}]\cdots]_{p_i}\sigma$$

In particular, if $t_{n+1} = t_1$, then

$$\mathcal{C}_1,\ldots,\mathcal{C}_n \models \mathcal{D} \vee (t_j \approx \Gamma_j[\cdots[\Gamma_n[\Gamma_1[\cdots[\Gamma_{i-1}[t_i]_{p_{i-1}}]\cdots]_{p_1}]_{p_n}]\cdots]_{p_j})\sigma$$

Proof. The proof follows immediately from the definition, by transitivity and substitutivity of $\approx$. □

Proposition 22. *Let $\bar{t} = (t_1,\ldots,t_n,t_1)$ be a cycle. For any number k, the sequence $\bar{s} = (t_{1+k},\ldots,t_{n+k},t_{1+k})$, with $t_i := t_{i-n}$ if $i > n$, is a cycle.*

Proof. Let $(\mathcal{C}_1,\ldots,\mathcal{C}_n)$ be the sequence of clauses forming $\bar{t}$. It is easy to check that $\bar{s}$ is a cycle formed by $(\mathcal{C}_{1+k},\ldots,\mathcal{C}_{n+k})$ with $\mathcal{C}_i := \mathcal{C}_{i-n}$ if $i > n$, as conditions of Definition 18 are invariant by circular permutation if $t_1 = t_{n+1}$. □

Definition 23. A chain $(t_1,\ldots,t_{n+1})$ built on a clause sequence $(\mathcal{C}_1,\ldots,\mathcal{C}_n)$ is an *extension* of an acyclic chain $(s_1,\ldots,s_{m+1})$ if $n \geq m$, the latter chain is built on $(\mathcal{C}_1,\ldots,\mathcal{C}_m)$, and the same (oriented) literals and positions are considered in each clause $\mathcal{C}_i$ in both chains.

Proposition 24. *Let $\bar{s}$ be a chain of constructor context $\Gamma[\bullet]_p$ and of mgu σ, and let $\bar{t}$ be an extension of $\bar{s}$. Then the mgu of $\bar{t}$ is of the form $\sigma\theta$, $\bar{t}$ is of the form $(\bar{s}\theta,\bar{u})$, and the constructor context of $\bar{t}$ is of the form $\Gamma\theta[\Delta[\bullet]_q]_p$.*

Proof. It is clear that the unification problem associated with $\bar{s}$ (as defined in Definition 18, condition 2) is included into that of $\bar{t}$. Hence the mgu of $\bar{t}$ is an instance of that of $\bar{s}$. Then the proof follows immediately from the definitions. The case where $\bar{u}$ is empty occurs when $\bar{s} = \bar{t}$, or when $m = n$ and $\bar{t}$ is a cycle (i.e., $\bar{t}$ is built on the same clauses as $\bar{s}$, but it satisfies condition 2.2 of Definition 18 instead of 2(a)). □

Since chains can be arbitrarily long, we need to impose some additional conditions to prune them and ensure that the rules are finitely branching. Let **Keep** be a property of chains that fulfills the following requirements:

(i) if a chain $\bar{t}$ does not satisfy **Keep**, no extension of $\bar{t}$ satisfies **Keep**;
(ii) for every finite clause set N, the set of chains built on a sequence of renamings of clauses in N and satisfying **Keep** is finite;
(iii) for every cycle $(t_1,\ldots,t_n,t_1)$, there exists a chain $(s_1,\ldots,s_m)$ with $m \leq n$ satisfying **Keep** such that for some k, the cycle $(t_{1+k},\ldots,t_{n+k},t_{1+k})$ (with $t_i := t_{i-n}$ if $i > n$) is an extension of $(s_1,\ldots,s_m)$.

For example, **Keep** can be defined as the set of chains built on clauses $\mathcal{C}_i$ that are pairwise distinct modulo renaming and such that $\mathcal{C}_1$ is the most recently processed clause. This is the definition we use in our description of the extended saturation loop (Section 6) and in the implementation in Vampire.

Remark 25. Condition (i) is essential in practice, to ensure that the chains can be incrementally constructed in an efficient way, because it ensures that the construction can be stopped when a prefix not satisfying **Keep** is obtained. Condition (ii) is not used in the following, but it ensures that the rule is finitely branching. Condition (iii) is essential for completeness.

Definition 26. A chain of length n is *eligible* if it is variable-ended and $n = 1$, or if it is not variable-ended, it satisfies **Keep**, and either it is a cycle or there exists an extension of length $n+1$ that does not satisfy **Keep**.

Remark 27. The conditions on eligible chains are the strongest ones preserving completeness, but they are not necessary for soundness. They may thus freely be relaxed if this yields a more efficient procedure.

The acyclicity rule follows:

$$\frac{\mathcal{C}_1 \quad \cdots \quad \mathcal{C}_n}{\mathcal{D} \vee \mathcal{E}}\ \textsf{Acycl}$$

if there exists a direct, eligible chain $(t_1,\ldots,t_{n+1})$ built on $(\mathcal{C}_1,\ldots,\mathcal{C}_n)$ under condition $\mathcal{D}$ and either $t_1 = t_{n+1}$ and $\mathcal{E} = \varnothing$ or $t_1 \neq t_{n+1}$ and $\mathcal{E} = \neg\mathsf{sub}(t_1,t_{n+1})$

Intuitively, the existence of the chain guarantees (if $\mathcal{D}$ is false) that there exists a nonempty constructor context $\Gamma[\bullet]_p$ such that $t_1 \approx \Gamma[t_{n+1}]_p$ holds. If $t_1 = t_{n+1}$, this contradicts acyclicity. Otherwise, we deduce that t_1 cannot occur at a constructor position inside the constructor term corresponding to t_{n+1}; hence $\mathsf{sub}(t_1,t_{n+1})$ is false. Since the interpretation of sub is not completely axiomatized, the Acycl rule is not sound in general, in the sense that the derived clauses are not logical consequences of the premises, even if the considered interpretation satisfies condition **Acy**. However, it is sound if one restricts oneself to interpretations that are sub-minimal. By Lemma 10, we know that this restriction does not involve any loss of generality, because a sub-minimal model exists for every satisfiable clause set (with no occurrences of sub).

Lemma 28 (Soundness of Acycl). *Let N be a clause set, and let $\mathcal{I}$ be a* sub*-minimal model of N satisfying* **Acy**. *If $\mathcal{C}$ is derived from a clause set N by* Acycl*, then $\mathcal{I} \models \mathcal{C}$.*

Proof. Let η be a valuation, and let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models N$ and $\mathcal{I},\eta \not\models \mathcal{D} \vee \mathcal{E}$ (with the notations of the rule). By Proposition 21, we have $\mathcal{I},\eta \models t_1 \approx \Gamma[t_{n+1}]_p$ for some nonempty constructor context. If $t_1 = t_{n+1}$, this yields an immediate contradiction with the hypothesis that $\mathcal{I}$ satisfies **Acy**. Otherwise, we must have $\mathcal{E} = \neg\mathsf{sub}(t_1,t_{n+1})$, hence $\mathcal{I},\eta \models \mathsf{sub}(t_1,t_{n+1})$. Because $\mathcal{I}$ is sub-minimal, this entails that there exists a constructor context $\Delta[\bullet]_q$ such that $\mathcal{I},\eta \models \exists\bar{z}.\ t_{n+1} \approx \Delta[t_1]_q$, thus $\mathcal{I},\eta \models \exists\bar{z}.\ t_{n+1} \approx \Delta[\Gamma[t_{n+1}]_p]_q$. Again, this contradicts the hypothesis that $\mathcal{I}$ satisfies **Acy**. □

4.6 Uniqueness of Fixpoints

The uniqueness rule also depends on the notion of chain:

$$\frac{\mathcal{C}_1 \quad \cdots \quad \mathcal{C}_n}{\mathcal{D} \vee \left(\bigvee_{p\in P} u|_p \not\approx \mathsf{app}(s_p,t_1)\right) \vee u' \not\approx z \vee z \approx t_1}\ \mathsf{Uniq}$$

if there exists an eligible chain $(t_1,\ldots,t_{n+1})$ of constructor context $\Gamma[\bullet]_q$ built on $(\mathcal{C}_1, \ldots,\mathcal{C}_n)$ under condition $\mathcal{D}$ and the following requirements are met:

1. $u = \Gamma[t_{n+1}]_q$;
2. P is the set of positions p of some type $\tau \sim \mathrm{type}(t_1)$ in u such that $p \not< q$;
3. for every $p \in P$, s_p is a fresh variable of type $[\upsilon]_\tau$, where υ,τ are the types of $u|_p$ and t_1, respectively;
4. u' is obtained from u by replacing all terms at a position $p \in P$ by $\mathsf{app}(s_p,z)$.

Intuitively, the existence of the chain ensures (if $\mathcal{D}$ is false) that $t_1 \approx \Gamma[t_{n+1}]_q$. If $t_1 = t_{n+1}$, we could derive $y \not\approx \Gamma[y]_q \vee y \approx t_1$ by uniqueness. However, this would not be sufficient for completeness. First, t_1 may be distinct from t_{n+1}, but we may have $t_{n+1} = \Delta[t_1]_Q$, for some constructor context Δ, in which case we should derive $y \not\approx \Gamma[\Delta[y]_Q]_q \vee y \approx t_1$ instead. Second, t_1 may also occur at other positions in Γ (not $<$-comparable with q). To capture all these cases using a finitely branching rule (i.e., without having to "guess" constructor contexts), we introduce new variables s_p whose purpose is to denote the context Γ_p such that $\Gamma_p[t_1] = u|_p$. (If t_1 does not occur inside $u|_p$, then Γ_p is constant.)

Example 29. From the clause $\mathsf{a} \approx \mathsf{c}(\mathsf{b}, x)$, using the chain (a, x), with the constructor context $\mathsf{c}(\mathsf{b}, \bullet)$, we derive

$$\mathsf{b} \not\approx \mathsf{app}(x_1, \mathsf{a}) \vee x \not\approx \mathsf{app}(x_2, \mathsf{a}) \vee z \not\approx \mathsf{c}(\mathsf{app}(x_1, z), \mathsf{app}(x_2, z)) \vee z \approx \mathsf{a}$$

Then $u = \mathsf{c}(\mathsf{b}, x)$ and $P = \{1, 2\}$.

From the clauses $\mathsf{a} \approx \mathsf{c}(\mathsf{b}, \mathsf{a})$ and $\mathsf{b} \approx \mathsf{d}(\mathsf{a}, \mathsf{a})$, using the chain $(\mathsf{a}, \mathsf{b}, \mathsf{a})$, with the constructor context $\mathsf{c}(\mathsf{d}(\mathsf{a}, \bullet), \mathsf{a})$, we derive

$$\begin{aligned} &\mathsf{a} \not\approx \mathsf{app}(x_{1.1}, \mathsf{a}) \vee \mathsf{a} \not\approx \mathsf{app}(x_{1.2}, \mathsf{a}) \vee \mathsf{a} \not\approx \mathsf{app}(x_2, \mathsf{a}) \\ &\vee z \not\approx \mathsf{c}(\mathsf{c}(\mathsf{app}(x_{1.1}, z), \mathsf{app}(x_{1.2}, z)), \mathsf{app}(x_2, z)) \vee z \approx \mathsf{a} \end{aligned}$$

In this case, $u = \mathsf{c}(\mathsf{d}(\mathsf{a}, \mathsf{a}), x)$ and $P = \{1.1, 1.2, 2\}$.

Lemma 30 (Soundness of Uniq). *Let N be a clause set, and let $\mathcal{I}$ be a model of $N \cup \{\mathsf{App}, \mathsf{Hole}\}$ satisfying* **FP**. *If $\mathcal{C}$ is derived from N by* Uniq, *then $\mathcal{I} \models \mathcal{C}$.*

Proof. Note that since $\mathcal{I}$ satisfies **FP**, necessarily $\mathcal{I} \models \mathsf{Uniq}$. Let η be a valuation. By Proposition 21, we have $\mathcal{I}, \eta \models \mathcal{D} \vee t_1 \approx u$. Let $\boxed{u}$ be the term obtained from u by replacing each constructor c by $\boxed{\mathsf{c}}_\tau$ and each term at a position $p \in P$ by s_p. It is straightforward to verify (by induction on u) that $\mathsf{App} \models \mathsf{app}(\boxed{u}, z) \approx u'$ and that

$$\mathsf{App} \models \left(\textstyle\bigwedge_{p \in P} u|_p \approx \mathsf{app}(s_p, t_1)\right) \Rightarrow u \approx \mathsf{app}(\boxed{u}, t_1)$$

Consequently, $\mathcal{I}, \eta \models \mathcal{D} \vee \bigvee_{p \in P} u|_p \not\approx \mathsf{app}(s_p, t_1) \vee t_1 \approx \mathsf{app}(\boxed{u}, t_1)$. Furthermore, we have $\mathsf{Hole} \models \boxed{u} \not\approx \mathsf{hole}$, because by definition of chains $\Gamma[\bullet]_P$ is not empty. Since

$$\mathsf{Uniq} \models \boxed{u} \approx \mathsf{hole} \vee t_1 \not\approx \mathsf{app}(\boxed{u}, t_1) \vee z \not\approx \mathsf{app}(\boxed{u}, z) \vee z \approx t_1$$

we deduce that

$$\mathcal{I}, \eta \models \mathcal{D} \vee \textstyle\bigvee_{p \in P} u|_p \not\approx \mathsf{app}(s_p, t_1) \vee u' \not\approx z \vee z \approx t_1$$

□

We also introduce the following optional simplification rule:

$$\frac{\Gamma_n[\cdots[\Gamma_1[s']_P]\cdots]_P \approx s \vee \mathcal{C}}{(\Gamma_1[s]_P \approx s \vee \mathcal{C})\sigma}\ \mathsf{Compr}$$

where s and s' are terms of the same type $\tau \in \mathcal{T}_{\mathsf{coind}}$ and P is a nonempty set of constructor position in Γ_i, for $i \in [1, n]$, such that $\varepsilon \notin P$, and $\sigma = \mathrm{mgu}\{s \stackrel{?}{=} s', \Gamma_1 \stackrel{?}{=} \cdots \stackrel{?}{=} \Gamma_n\}$.

Proposition 31 (Soundness of Compr). *Let N be a clause set, and let $\mathcal{I}$ be a model of N satisfying* **FP**. *If $\mathcal{D}$ is derived from N by* Compr, *then $\mathcal{I} \models \mathcal{D}$.*

Proof. Let $\mathcal{D} = (\Gamma_1[s]_P \approx s \vee \mathcal{C})\sigma$, and let η be a valuation such that $\mathcal{I}, \eta \not\models \mathcal{C}\sigma$. Since $\mathcal{I} \models N$, we have $\mathcal{I}, \eta \models (\Gamma_n[\cdots[\Gamma_1[s']_P]\cdots]_P \approx s)\sigma$, i.e., $\mathcal{I}, \eta \models (\Gamma_n[\cdots[\Gamma_1[s]_P]\cdots]_P \approx s)\sigma$. Then, since $\mathcal{I}$ satisfies **FP** (existence of fixpoints), $\mathcal{I}, \eta \models \exists x.\ x \approx (\Gamma_1\sigma)[x]_P$, thus there exists an extension η' of η such that $\mathcal{I}, \eta' \models x \approx (\Gamma_1\sigma)[x]_P$ for some fresh variable x. Then, since $\Gamma_1\sigma = \cdots = \Gamma_n\sigma$, we have $\mathcal{I}, \eta' \models x \approx \Gamma_n[\cdots[\Gamma_1[x]_P]\cdots]_P$, and since $\mathcal{I}$ satisfies **FP** (uniqueness) we deduce that $\mathcal{I}, \eta' \models s\sigma \approx x$ thus $\mathcal{I}, \eta \models (\Gamma_1[s]_P \approx s)\sigma$. □

5 Refutational Completeness

We establish the refutational completeness of the calculus presented in Section 4. This result ensures that the axioms for distinctness, injectivity, and acyclicity (NSub) may be omitted. The axiom Uniq may also be omitted in some cases, formally defined below. The axiom Sub is still needed since it is used in the completeness proof for Acycl.

If $N \not\ni \square$ is a clause set saturated under $\mathcal{SP}$, then R_N denotes the set of rewrite rules constructed as usual from N and $\to_{R_N}$ denotes the (one-step) reduction relation. We refer to the literature [2, 15] for details about the construction of R_N. The notation $\mathcal{M}_N$ denotes the model of N defined by the congruence $\overset{*}{\leftrightarrow}_{R_N}$ on ground terms.

We first establish some results about the form of the rules in R_N.

Proposition 32. *Let N be a clause set saturated under $\mathcal{SP}$ and* Inf. *Let $u \approx v \vee \mathcal{C} \in N$, and let θ be a substitution such that $u\theta \succ v\theta$, $(u \approx v)\theta \succ \mathcal{C}\theta$, and $\mathcal{M}_N \not\models \mathcal{C}\theta$. If* $\mathrm{type}(u) \in \mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$, *then u is not a variable.*

Proof. If u is a variable, then due to the order conditions u cannot occur in the scope of a function symbol in $\mathcal{C}$ or v, or in a negative literal of $\mathcal{C}$, hence it occurs only at root positions in equations. Consequently, $u \approx v \vee \mathcal{C}$ is of the form $\bigvee_{i=1}^{n} u \approx t_i \vee \mathcal{C}'$, where u does not occur in $\mathcal{C}'$ or $t_1, \ldots, t_n$ (with $v \in \{t_1, \ldots, t_n\}$). Then Inf applies and derives $\mathcal{C}'$. Since N is saturated under Inf, we deduce that $\mathcal{M}_N \models \mathcal{C}' \models \mathcal{C}\theta$, which contradicts the hypotheses. □

Corollary 33. *Let N be a clause set saturated under $\mathcal{SP}$ and* Inf. *For every rule* $\mathsf{c}(\bar{t}) \to_{R_N} s$ *in R_N, where* c *is a constructor, s is R_N-irreducible.*

Proof. Assume that s is reducible in R_N. This means that there exist a subterm u at position p of s and a rule $u \to_{R_N} v$ in R_N. Consequently, there exist two clauses $\mathcal{C} = t' \approx s' \vee \mathcal{C}'$ and $\mathcal{D} = u' \approx v' \vee \mathcal{D}'$, and two substitutions σ and θ such that $t'\theta = \mathsf{c}(\bar{t})$, $u'\sigma = u$, $s'\theta = s$, $v'\sigma = v$, and $\mathcal{M}_N \not\models \mathcal{C}'\theta \vee \mathcal{D}'\sigma$. By definition of R_N, $x\theta$ is R_N-irreducible, for every $x \in \mathrm{dom}(\theta)$, consequently p is a non variable position in s'. By Proposition 32, u' is not a variable, hence the head symbol of t' is c. Therefore, superposition into s' is allowed in the relaxed calculus, and the inference yields a clause $(t' \approx s'[v']_p \vee \mathcal{C}' \vee \mathcal{D}')\eta$, where $\eta = \mathrm{mgu}\{s'|_p \stackrel{?}{=} u'\}$. Since $\theta\sigma$ is an instance of η, we deduce that there exist ground clauses $\mathcal{E}_1, \ldots, \mathcal{E}_n$ that are instances of clauses in N such that $\mathsf{c}(\bar{t}) \approx s[v]_p \vee \mathcal{C}'\theta \vee \mathcal{D}'\sigma \succeq \mathcal{E}_i$ for $i \in [1, n]$ and $\mathcal{E}_1, \ldots, \mathcal{E}_n \models \mathsf{c}(\bar{t}) \approx s[v]_p \vee \mathcal{C}'\theta \vee \mathcal{D}'\sigma$. Let R'_N be the rules added before $\mathsf{c}(\bar{t}) \to_{R_N} s$ in R_N and let $\mathcal{M}'$ be the corresponding interpretation. By construction of the model, $\mathsf{c}(\bar{t})$ is R'_N-irreducible and $\mathcal{M}' \not\models \mathcal{C}'\theta \vee \mathcal{D}'\sigma$, moreover, since $\mathcal{C}\theta \succ \mathcal{E}_i$, we have $\mathcal{M}' \models \mathcal{E}_1, \ldots, \mathcal{E}_n$ thus $\mathcal{M}' \models \mathsf{c}(\bar{t}) \approx s[v]_p$, a contradiction. □

Lemma 34 (Infiniteness). *Let N be a clause set saturated under $\mathcal{SP}$ and* Inf. *If $\square \notin N$, then $\mathcal{M}_N$ satisfies* **Inf**.

Proof. Let $\tau \in \mathcal{T}_{\mathsf{ind}} \cup \mathcal{T}_{\mathsf{coind}}$. We assume, without loss of generality, that the model is constructed over a signature that contains at least two (non-constructor) symbols $\mathsf{a} : \tau$ and $\mathsf{f} : \tau \to \tau$, not occurring in N. By Proposition 32, all the rules in R_N are of the form $\mathsf{g}(\bar{t}) \to_{R_N} s$, where g occurs in N. Thus $\mathsf{f}^n(\mathsf{a})$ is R_N-irreducible for every natural number n, and the domain of τ is infinite. □

Lemma 35 (Distinctness). *Let N be a clause set saturated under $\mathcal{SP}$, Dist_1, Dist_2, and Inf. For all ground terms $a = \mathsf{c}(\bar{a})$ and $b = \mathsf{d}(\bar{b})$ such that $\mathsf{c} \bowtie \mathsf{d}$, we have $a \overset{*}{\not\leftrightarrow}_{R_N} b$.*

Proof. Assume $a \overset{*}{\leftrightarrow}_{R_N} b$. Without loss of generality, we can assume that $a \succ b$ and that $\bar{a}$ and $\bar{b}$ are R_N-irreducible. The proof proceeds by case analysis over the reducibility of a and b.

- a is irreducible. Since $a \succ b$, it cannot be the case that $a \overset{*}{\leftrightarrow}_{R_N} b$.
- b is irreducible and $a \overset{+}{\rightarrow}_{R_N} b$. Then R_N contains a rule $a \rightarrow_{R_N} a'$, with $a' \rightarrow^*_{R_N} b$. By Corollary 33, we know that a' is R_N-irreducible, therefore we must have $a' = b$. There exist a clause $\mathcal{C} = u \approx v \vee \mathcal{C}'$ and a substitution θ, with $\mathcal{M}_N \not\models \mathcal{C}'\theta$, $u\theta = a$, and $v\theta = b$. By Proposition 32, u cannot be a variable, hence its head symbol is c. Consequently, there is an inference Dist_1 taking $\mathcal{C}$ for premise, and deriving a clause $\mathcal{C}'\sigma$, with $\sigma = \mathrm{mgu}\{v \stackrel{?}{=} \mathsf{c}(\bar{x})\}$. It is clear that θ is an instance of σ (more exactly of the restriction of σ to the variables of $\mathcal{C}$). Thus $\mathcal{M}_N \models \mathcal{C}'\sigma \models \mathcal{C}'\theta$, leading to a contradiction.
- There exists a term c such that $a \overset{+}{\rightarrow}_{R_N} c$ and $b \overset{+}{\rightarrow}_{R_N} c$. Then R_N must contain two rules of the form $a \rightarrow a'$ and $b \rightarrow b'$, corresponding to two clauses $\mathcal{C}$ and $\mathcal{D}$ of the form $u \approx u' \vee \mathcal{C}'$ and $v \approx v' \vee \mathcal{D}'$ with $\mathcal{M}_N \not\models \mathcal{C}\theta \vee \mathcal{C}'\theta'$, $a' \overset{*}{\rightarrow}_{R_N} c$, $b' \overset{*}{\rightarrow}_{R_N} c$, $u\theta = a$, $v\theta' = b$, $u'\theta = a'$, and $v'\theta' = b'$. By Proposition 32, u and v cannot be variables, hence there head symbols must be c and d respectively. By Corollary 33, a' and b' are R_N-irreducible, and therefore $a' = b' = c$. The rule Dist_2 can be applied to clauses $\mathcal{C}$ and $\mathcal{D}$, yielding a clause $(\mathcal{C} \vee \mathcal{C}')\sigma$, where $\sigma = \mathrm{mgu}\{u' \stackrel{?}{=} v'\}$. Then $\theta\theta'$ is an instance of σ, thus $\mathcal{M}_N \models (\mathcal{C} \vee \mathcal{C}')\sigma \models \mathcal{C}\theta \vee \mathcal{C}'\theta'$, a contradiction.

□

Lemma 36 (Injectivity). *Let N be a clause set saturated under $\mathcal{SP}$, Inf, Inj_1, and Inj_2. For all ground terms $a = \mathsf{c}(a_1, \ldots, a_n)$ and $b = \mathsf{c}(b_1, \ldots, b_n)$ with $\mathsf{c} \in \mathcal{C}tr_{\mathsf{inj}}$ and such that $a_i \overset{*}{\not\leftrightarrow}_{R_N} b_i$ for some $i \in [1, n]$, we have $a \overset{*}{\not\leftrightarrow}_{R_N} b$.*

Proof. Assume $a \overset{*}{\leftrightarrow}_{R_N} b$. Without loss of generality, we can assume that $a \succ b$ and that $a_1, \ldots, a_n, b_1, \ldots, b_n$ are R_N-irreducible. The proof is similar to that of Lemma 35.

- a is irreducible. Since $a \succ b$, it cannot be the case that $a \overset{*}{\leftrightarrow}_{R_N} b$.
- b is irreducible and $a \overset{+}{\rightarrow}_{R_N} b$. Then R_N contains a rule $a \rightarrow_{R_N} a'$, with $a' \rightarrow^*_{R_N} b$. By Corollary 33, we know that a' is irreducible, therefore we must have $a' = b$. There exists a clause $\mathcal{C} = u \approx v \vee \mathcal{C}'$, with $\mathcal{M}_N \not\models \mathcal{C}'\theta$, $u\theta = a$, and $v\theta = b$. By Proposition 32, u cannot be a variable, thus u is of the form $\mathsf{c}(u_1, \ldots, u_n)$. Then there is an inference Dist_1 taking $\mathcal{C}$ for premise, and deriving a clause $(u_i \approx x_i \vee \mathcal{C}')\sigma$, with σ is an mgu of v and $\mathsf{c}(x_1, \ldots, x_n)$. The substitution $\{x_i \mapsto b_i\}\theta$ is an instance of σ, thus $\mathcal{M}_N \models (u_i \approx x_i \vee \mathcal{C}')\sigma \models (u_i \approx x_i \vee \mathcal{C}')\{x_i \mapsto b_i\}\theta = a_i \approx b_i \vee \mathcal{C}'\theta$, leading to a contradiction.
- There exists a term c such that $a \overset{+}{\rightarrow}_{R_N} c$ and $b \overset{+}{\rightarrow}_{R_N} c$. Then R_N must contain two rules of the form $a \rightarrow a'$ and $b \rightarrow b'$, corresponding to two clauses $\mathcal{C}$ and $\mathcal{D}$ of the form $u \approx u' \vee \mathcal{C}'$ and $v \approx v' \vee \mathcal{D}'$ with $\mathcal{M}_N \not\models \mathcal{C}\theta \vee \mathcal{C}'\theta'$, $a' \overset{*}{\rightarrow}_{R_N} c$, $b' \overset{*}{\rightarrow}_{R_N} c$, $u\theta = a$, $v\theta' = b$, $u'\theta = a'$, and $v'\theta' = b'$. By Proposition 32, u and v cannot be

variables, hence they must be of the form $\mathsf{c}(u_1,\ldots,u_n)$ and $\mathsf{c}(v_1,\ldots,v_n)$ respectively. By Corollary 33, a' and b' are R_N-irreducible, and therefore $a' = b' = c$. The rule Dist_2 can be applied to clauses $\mathcal{C}$ and $\mathcal{D}$, yielding a clause of the form $(u_i \approx v_i \vee \mathcal{C} \vee \mathcal{C}')\sigma$, where $\theta\theta'$ is an instance of σ. Thus $\mathcal{M}_N \models (u_i \approx v_i \vee \mathcal{C} \vee \mathcal{C}')\sigma \models a_i \approx b_i \vee \mathcal{C}\theta \vee \mathcal{C}'\theta'$, a contradiction.

□

The completeness proof for acyclicity requires further definitions and results.

Definition 37. Let $\mathcal{I}$ be an interpretation and t be a term. A constructor context $\Gamma[\bullet]_p$ is a *minimal cyclicity witness* for t and $\mathcal{I}$ if p is a position of the same type as t in Γ, $\mathcal{I} \models t \approx \Gamma[t]_p$, and $|q| \geq |p|$ for every position $q \neq \varepsilon$ and constructor context $\Delta[\bullet]_q$ such that $\mathcal{I} \models t \approx \Delta[t]_q$.

Proposition 38. *Let $(t_1,\ldots,t_n,t_1)$ be a cycle of constructor context $\Gamma[\bullet]_p$ for a clause set N under condition $\mathcal{D}$. If $\mathcal{I} \models N \cup \{\neg\mathcal{D}\sigma\}$, and $\Gamma[\bullet]_p$ is a minimal cyclicity witness for $t_1\sigma$ and $\mathcal{I}$, then $(t_1,\ldots,t_n,t_1)$ is direct.*

Proof. Assume that $(t_1,\ldots,t_n,t_1)$ is not direct. By definition, there exist $i,j \in [1,n]$ such that $i < j$ and $t_i = t_j$. By Proposition 21, since $\mathcal{I} \not\models \mathcal{D}\sigma$ we have $\mathcal{I} \models (t_1 \approx \Gamma[t_1]_{p_1\cdots.p_n})\sigma$ for some positions $p_1,\ldots,p_n$, with $p = p_1.\ldots.p_n$. Furthermore, again by Proposition 21, we also have $\mathcal{I} \models (t_1 \approx \Gamma_i[t_i]_{p_1\cdots.p_{i-1}})\sigma$ and $\mathcal{I} \models (t_j \approx \Gamma_j[t_n]_{p_j\cdots.p_{n-1}})\sigma$, for some constructor contexts Γ_i and Γ_j.

Since $t_i = t_j$, we deduce: $\mathcal{I} \models (t_1 \approx \Gamma_i[\Gamma_j[t_1]_{p_j.\cdots.p_n}]_{p_1.\cdots.p_{i-1}})\sigma$, which entails that $\Gamma\sigma$ is not a minimal cyclicity witness for $t_1\sigma$, because, since $i < j$, necessarily $0 < \left|p_1.\cdots.p_{i-1}.p_j.\cdots.p_n\right| < |p_1.\cdots.p_n|$. □

Lemma 39. *Let $t : \tau$ and $s : \upsilon$ be ground terms with $\tau \sim \upsilon$. Let $\Gamma[\bullet]_p$ be a ground constructor context of type τ, where p is a position of type υ in Γ. Let N be a clause set saturated under $\mathcal{SP}$ and* Inf. *Assume that t, s, and $\Gamma|_{p'}$ are R_N-irreducible, for every position $p' \not\leq p$. If $\mathcal{M}_N \models \Gamma[s]_p \approx t$, then R_N contains n rules $\Gamma_i[a_{i+1}]_{p_i} \to_{R_N} a_i$, for $i \in [1,n]$, with $\Gamma[s]_p = \Gamma_0[\Gamma_1[\cdots[\Gamma_n[a_{n+1}]_{p_n}]\cdots]_{p_1}]_{p_0}$, $p_0.p_1.\cdots.p_n = p$, $a_{n+1} = s$, and $t = \Gamma_0[a_1]_{p_0}$.*

Proof. Let $t' = \Gamma[s]_p$. Since $\mathcal{M}_N \models t' \approx t$ and t is R_N-irreducible, we have $t' \to_{R_N}^k t$, for some natural number $k \geq 0$. The proof is by induction on k. It is immediate if $k = 0$, since in this case $t = t'$, by letting $n = 1$, $\Gamma_0 = \Gamma$ and $p_0 = p$ (with $a_1 = s$). Otherwise, since $\Gamma|_{p'}$ is R_N-irreducible, for $p' \not\leq p$, the first rule in the $\to_{R_N}$-derivation from t' to t must be applied at some position $p' \leq p$. We denote by p_n the position such that $p = p'.p_n$. Thus there exists a rule $\Delta[s]_{p_n} \to_{R_N} b$ such that $t'|_{p'} = \Delta[s]_{p_n}$, and $t'[b]_{p'} \to_{R_N}^{k-1} t$. Since Γ is a constructor context, the head symbol of Δ is a constructor, and by Corollary 33, b must be R_N-irreducible. By the induction hypothesis, since $t'[b]_{p'} \to_{R_N}^{k-1} t$, there exist rules $\Gamma_i[a_{i+1}]_{p_i} \to_{R_N} a_i$, for $i \in [1,n-1]$, with $t'[b]_{p'} = \Gamma_0[\cdots[\Gamma_{n-1}[a_n]_{p_{n-1}}]\cdots]_{p_0}$, $p_0.\cdots.p_{n-1} = p'$, $a_n = b$ and $t = \Gamma_0[a_1]_{p_0}$. By letting $\Gamma_n = \Delta$ and $a_{n+1} = s$, we get $t' = \Gamma_0[\cdots[\Gamma_{n-1}[\Gamma_n[a_{n+1}]_{p_n}]_{p_{n-1}}]\cdots]_{p_0}$ with $p_0.\cdots.p_{n-1}.p_n = p$. □

Lemma 40 (Acyclicity). *If* $\mathsf{Sub} \subseteq N$ *and* $N \not\ni \square$ *is saturated under $\mathcal{SP}$,* Acycl, *and* Inf, *then $\mathcal{M}_N$ satisfies condition* **Acy**.

Proof. Assume that there exist a ground term t of some type $\tau \in \mathcal{T}_{\mathsf{ind}}$ and a ground constructor context $\Gamma[\bullet]_p$ such that $\mathcal{M}_N \models t \approx \Gamma[t]_p$, with $p \neq \varepsilon$. W.l.o.g., we may assume that t is a minimal term (with respect to $\succ$) of some type in $\mathcal{T}_{\mathsf{ind}}$ such that a context Γ satisfying the condition above exists, that Γ is a minimal cyclicity witness for t and $\mathcal{M}_N$ and that $\Gamma|_{p'}$ is R_N-irreducible for every $p' \not\leq p$. Since t is minimal, t is R_N-irreducible. Let $t' = \Gamma[t]_p$.

By Lemma 39, R_N contains n rules $\Gamma_i[a_{i+1}]_{p_i} \to_{R_N} a_i$, for $i \in [1,n]$, such that $t' = \Gamma_0[\cdots[\Gamma_n[a_{n+1}]_{p_n}]\cdots]_{p_0}$, $p_0.\cdots.p_n = p$, $a_{n+1} = t$ and $t = \Gamma_0[a_1]_{p_0}$. If $p_0 \neq \varepsilon$, then $t \succ a_1$, and $\mathcal{M}_N \models a_1 \approx \Gamma_1[\cdots[\Gamma_n[\Gamma_0[a_1]_{p_0}]]_{p_n}]\cdots]_{p_0}$, which contradicts the minimality of t, because $\mathrm{type}(a_1) \sim \mathrm{type}(t) \in \mathcal{T}_{\mathsf{ind}}$. Thus we have $p_0 = \varepsilon$ and Γ_0 is empty.

By definition of R_N, there exist n clauses $\mathcal{C}_i = u_i \approx s_i \vee \mathcal{C}'_i$ and substitutions θ_i (for $i \in [1,n]$) such that, for every $i \in [1,n]$, $u_i\theta_i \succ s_i\theta_i$, $(u_i \approx s_i)\theta_i \succ \mathcal{C}'_i\theta_i$, $u_i\theta_i = \Gamma_i[a_{i+1}]_{p_i}$, $s_i\theta_i = a_i$ and $\mathcal{M}_N \not\models \mathcal{C}'_i\theta_i$. Let q_i be the maximal prefix of p_i that is a position in u_i. We distinguish two cases.

- If there exists $i \in [1,n]$ such that $u_i|_{q_i}$ is a variable x, then $\mathcal{C}_i$ must be of the form $u_i[x]_{q_i} \approx s_i \vee \mathcal{C}'_i$. By Proposition 32, u_i is not a variable, hence $q_i \neq \varepsilon$. Consequently, the Acycl rule applies on $\mathcal{C}_i$ (using a trivial chain (s_i, x) of length 1 that is eligible since it ends with a variable) and derives the clause: $\mathcal{C}'_i \vee \neg\mathsf{sub}(s_i, x)$. Since $\mathcal{M}_N$ is saturated under Acycl, this entails that $\mathcal{M}_N \models \mathcal{C}'_i \vee \neg\mathsf{sub}(s_i, x)$, hence $\mathcal{M}_N \not\models \mathsf{sub}(s_i, x)\theta_i$. Let q'_i be the position such that $q_i.q'_i = p_i$. We have $x = u_i|_{q_i}$, thus:

$$\begin{aligned} x\theta_i &\overset{*}{\leftrightarrow}_{R_N} \Gamma_i|_{q_i}[a_{i+1}]_{q'_i} \\ &\overset{*}{\leftrightarrow}_{R_N} \Gamma_i|_{q_i}[\Gamma_{i+1}[\cdots[\Gamma_n[a_{n+1}]_{p_n}]\cdots]_{p_{i+1}}]_{q'_i} \\ &\overset{*}{\leftrightarrow}_{R_N} \Gamma_i|_{q_i}[\Gamma_{i+1}[\cdots[\Gamma_n[t]_{p_n}]\cdots]_{p_{i+1}}]_{q'_i} \\ &\overset{*}{\leftrightarrow}_{R_N} \Gamma_i|_{q_i}[\Gamma_{i+1}[\cdots[\Gamma_{i-1}[\Gamma_1[\cdots[\Gamma_{i-1}[a_i]_{p_{i-1}}]\cdots]_{p_1}]_{p_{n-1}}]\cdots]_{p_{i+1}}]_{q'_i} \end{aligned}$$

  By Proposition 3, this entails that $\mathcal{M}_N \models \mathsf{sub}(a_i, x\theta_i)$, i.e., $\mathcal{M}_N \models \mathsf{sub}(s_i, x)\theta_i$, a contradiction.
- Otherwise, we must have $p_i = q_i$, for $i \in [1,n]$. Let $s'_{i+1} = u_i|_{p_i}$, for $i \in [1,n]$. The substitution $\theta_1 \ldots \theta_n$ is a solution of $s'_{n+1} \stackrel{?}{=} s_1 \wedge \bigwedge_{i=2}^{n} s'_i \stackrel{?}{=} s_i$, hence this problem admits an mgu σ, with $\theta_1 \ldots \theta_n = \sigma\eta$. Let $(t_1, \ldots, t_{n+1}) = (s_1, \ldots, s_{n+1})\sigma$. It is easy to check that all the conditions of Definition 18 are satisfied, hence $(t_1, \ldots, t_{n+1})$ is a cycle. By definition of **Keep**, there exists k such that $(t_{1+k}, \ldots, t_{n+k}, t_{1+k})$ (with $t_i := t_{i-n}$ if $i > n$) is a cycle, and there exists a chain $(b_1, \ldots, b_{m+1})$ satisfying **Keep** such that $(t_{1+k}, \ldots, t_{n+k}, t_{1+k})$ is an extension of $(b_1, \ldots, b_{m+1})$. We assume that $(b_1, \ldots, b_{m+1})$ is the longest chain with this property. By definition, this entails that $(b_1, \ldots, b_{m+1})$ is eligible (indeed, either $m = n$ and the chain is a cycle, or $(b_1, \ldots, b_{m+1}) \neq (t_{1+k}, \ldots, t_{n+k}, t_{1+k})$ and there exists an extension of length $m+2$ of $(b_1, \ldots, b_{m+1})$ that does not satisfied **Keep**). Furthermore, since Γ is a minimal cyclicity witness for t and $\mathcal{M}_N$, $(t_1, \ldots, t_{n+1})$ is direct by Proposition 38, hence $(b_1, \ldots, b_{m+1})$ is also direct. Consequently, the Acycl rule applies on the clauses $\mathcal{C}_{1+k}, \ldots, \mathcal{C}_{m+k}$, yielding a clause of the form $\mathcal{C}'_{1+k}\theta \vee \ldots \vee \mathcal{C}'_{m+k}\theta \vee \mathcal{E}$, where the subclause $\mathcal{E}$ is either $\Box$ or $\neg\mathsf{sub}(b_{m+1}, b_1)$. By Proposition 24, there exists σ' such that $\sigma = \theta\sigma'$ and $t_{i+k} = b_i\sigma'$ (for $i = 1, \ldots, m$). We have $\mathcal{C}'_i\theta\sigma'\eta = \mathcal{C}'_i\sigma\eta = \mathcal{C}_i\theta_i$, hence $\mathcal{M}_N \not\models \mathcal{C}'_{1+k}\theta\sigma'\eta \vee \ldots \vee \mathcal{C}'_{m+k}\theta\sigma'\eta$. Since N is saturated under

Acycl, $\mathcal{M}_N \models \mathcal{C}'_{1+k}\theta \vee \ldots \vee \mathcal{C}'_{m+k}\theta \vee \mathcal{E}$, hence we deduce that $\mathcal{M}_N \models \mathcal{E}\sigma'\eta$. Thus $\mathcal{M}_N \not\models \mathsf{sub}(b_{m+1}, b_1)\sigma'\eta$. By Proposition 21, there exist some constructor context Δ and position $r \neq \varepsilon$ such that $\mathcal{M}_N \models (t_{m+1+k} \approx \Delta[t_{1+k}]_r)\eta$, i.e., $\mathcal{M}_N \models (b_{m+1} \approx \Delta[b_1]_r)\sigma'\eta$. This contradicts the fact that $\mathcal{M}_N \not\models \mathsf{sub}(b_{m+1}, b_1)\sigma'\eta$.

□

Remark 41. The Inf rule is needed for completeness. For example, it is clear that the clause $x \approx \mathsf{a} \vee x \approx \mathsf{b}$ contradicts acyclicity, but no contradiction can be derived without using Inf. The relaxation of the application conditions of Sup is also essential. Consider the clause set $N = \{\mathsf{a}_1 \approx \mathsf{c}(\mathsf{a}_2), \mathsf{a}_2 \approx \mathsf{a}_3, \mathsf{a}_3 \approx \mathsf{c}(\mathsf{a}_1)\}$, with $\mathsf{c}(\ldots) \succ \mathsf{a}_{i+1} \succ \mathsf{a}_i$. It is clear that N is saturated without the relaxation, and N contradicts acyclicity, since $N \models \mathsf{a}_1 \approx \mathsf{c}(\mathsf{c}(\mathsf{a}_1))$. With the relaxation, Sup derives the clause $\mathsf{a}_2 \approx \mathsf{c}(\mathsf{a}_1)$; then Acycl exploits the cycle $(\mathsf{a}_1, \mathsf{a}_2, \mathsf{a}_1)$ to derive □.

For the Uniq rule, we provide a restricted completeness result, under the assumption that the considered constructor context contains at most one occurrence of $\bullet$.

Lemma 42 (Uniqueness of Fixpoints). *If* $\mathsf{App} \subseteq N$ *and* $N \not\ni \square$ *is saturated under* $\mathcal{SP}$, Uniq, *and* Inf, *then* $\mathcal{M}_N \models x \approx \Gamma[x]_r \wedge y \approx \Gamma[y]_r \Rightarrow x \approx y$ *for every constructor context of the form* $\Gamma[\bullet]_r$ *of type* $\tau \in \mathcal{T}_{\mathsf{coind}}$, *where* r *is a nonempty position of type* τ *in* Γ.

Proof. The proof is similar to that of Lemma 40. Let t be a ground term such that there exist a nonempty constructor context $\Gamma[\bullet]_r$ and a ground term s of type τ with $\mathcal{M}_N \models t \approx \Gamma[t]_r \wedge s \approx \Gamma[s]_r \wedge t \not\approx s$. W.l.o.g., we assume that t is a minimal (with respect to $\succ$) term such that Γ and s exist, and that $\Gamma|_p$ is irreducible, for every $p \not\leq r$. By Lemma 39, since $\mathcal{M}_N \models t \approx \Gamma[t]_r$, R_N contains n rules $\Gamma_i[a_{i+1}]_{p_i} \to_{R_N} a_i$, for $i \in [1,n]$, with $\Gamma[t]_r = \Gamma_0[\cdots[\Gamma_n[a_{n+1}]_{p_n}]\cdots]_{p_0}$, $p_0.\cdots.p_n = r$, $a_{n+1} = t$ and $t = \Gamma_0[a_1]_{p_0}$.

Assume first that $p_0 \neq \varepsilon$. We have:

$$\mathcal{M}_N \models a_1 \approx \Gamma_1[\cdots[\Gamma_n[\Gamma_0[a_1]_{p_0}]\ldots]_{p_n}]\cdots]_{p_0}$$

Moreover, from $\mathcal{M}_N \models s \approx \Gamma[s]_r$, we deduce:

$$\mathcal{M}_N \models \Gamma_1[\cdots[\Gamma_n[s]_{p_n}]\cdots]_{p_1} \approx \Gamma_1[\cdots[\Gamma_n[\Gamma[s]_r]_{p_n}]\cdots]_{p_1}$$

i.e.:

$$\mathcal{M}_N \models \Gamma_1[\cdots[\Gamma_n[s]_{p_n}]\cdots]_{p_1} \approx \Gamma_1[\cdots[\Gamma_n[\Gamma_0[\cdots[\Gamma_n[s]_{p_n}]\cdots]_{p_0}]_{p_n}]\cdots]_{p_1}$$

thus by letting $s' = \Gamma_1[\cdots[\Gamma_n[s]_{p_n}]\cdots]_{p_1}$, we get

$$\mathcal{M}_N \models s' \approx \Gamma_1[\cdots[\Gamma_{n-1}[\Gamma_0[s']_{p_0}]_{p_{n-1}}]\cdots]_{p_0}$$

By minimality of t, since $t \succ a_1$ we deduce that $\mathcal{M}_N \models a_1 \approx s'$, hence that $\mathcal{M}_N \models \Gamma_0[a_1]_{p_0} \approx \Gamma_0[s']_{p_0}$, i.e., $\mathcal{M}_N \models t \approx s$, which contradicts our assumption.

Thus $p_0 = \varepsilon$, and Γ_0 is empty. By definition of R_N, there exist clauses $\mathcal{C}_i = u_i \approx s_i \vee \mathcal{C}'_i$ and substitutions θ_i (for $i \in [1,n]$) such that, for every $i \in [1,n]$, $u_i\theta_i \succ s_i\theta_i$, $(u_i \approx s_i)\theta_i \succ \mathcal{C}'_i\theta_i$, $u_i\theta_i = \Gamma_i[a_{i+1}]_{p_i}$, $s_i\theta_i = a_i$ and $\mathcal{M}_N \not\models \mathcal{C}'_i\theta_i$. Let q_i be the maximal prefix of p_i that is a position in u_i. We distinguish two cases.

- If there exists $i \in [1,n]$ such that $u_i|_{q_i}$ is a variable y, then $\mathcal{C}_i$ must be of the form $u_i[y]_{q_i} \approx s_i \vee \mathcal{C}'_i$. We assume, w.l.o.g., that i is the minimal number having this property. By Proposition 32, u_i is not a variable, hence $q_i \neq \varepsilon$. Consequently, the Uniq rule applies on $\mathcal{C}_i$ and derives the clause $\mathcal{C}'_i \vee \mathcal{D}$, with

$$\mathcal{D} = \big(\bigvee_{p \in P} u_i|_p \not\approx \mathsf{app}(s_p, s_i)\big) \vee u' \not\approx z \vee z \approx s_i$$

where P is the set of positions p of some type $\tau \sim \text{type}(s_i)$ in $u_i[y]_{q_i}$ such that $p \not< q_i$. Since $\mathcal{M}_N$ is saturated under Acycl, this entails that $\mathcal{M}_N \models \mathcal{C}'_i \vee \mathcal{D}$, hence $\mathcal{M}_N \not\models \mathcal{D}\theta_i$. Γ can be written on the form

$$\Gamma = \Delta[\Delta'[\bullet]_{p_i.....p_n}]_{p_1.....p_{i-1}}$$

Every position $p \in P$ is a position in Γ_i, since $u_i\theta_i = \Gamma_i[a_{i+1}]_{p_i}$. Thus for every $p \in P$, $p_1.....p_{i-1}.p$ is a position in Γ. Moreover, by Proposition 8, there exists a term s'_p such that $\mathsf{App} \models \forall x.\ \mathsf{app}(s'_p, x) \approx \Gamma|_{p_1.....p_{i-1}.p}[\Delta[x]_{p_1.....p_{i-1}}]_{Q_p}$, where Q_p denotes the set of positions of $\bullet$ in $\Gamma|_{p_1.....p_{i-1}.p}$ (this set is either empty, if $p \neq q_i$, or a singleton if $p = q_i$). Let η be a substitution mapping each variable s_p to s'_p and mapping z to $\Delta'[s]_{p_i.....p_n}$. Observe that if $p \neq q_i$, then $\Gamma|_{p_1.....p_{i-1}.p}$ contains no occurrences of $\bullet$, hence $s_p\eta = \mathsf{cst}(\Gamma|_{p_1.....p_{i-1}.p}) = \mathsf{cst}(u_i|_p)\theta_i$. By definition of η, since $\mathcal{M}_N \models \mathsf{App}$, $\mathcal{M}_N \models (\mathsf{app}(s_p, s_i) \approx \Gamma|_{p_1.....p_{i-1}.p}[\Delta[s_i]_{p_1.....p_{i-1}}]_{Q_p})\theta_i\eta$. Furthermore, $\mathcal{M}_N \models s_i \approx \Delta'[t]_{p_i.....p_{n-1}}$ hence:

$$\mathcal{M}_N \models (\mathsf{app}(s_p, s_i) \approx \Gamma|_{p_1.....p_{i-1}.p}[\Delta[\Delta'[t]_{p_i.....p_n}]_{p_1.....p_{i-1}}]_{Q_p})\theta_i\eta$$

Thus we have $\mathcal{M}_N \models (\mathsf{app}(s_p, s_i) \approx \Gamma|_{p_1.....p_{i-1}.p}[t]_{Q_p})\theta_i\eta$ which entails that $\mathcal{M}_N \models (\mathsf{app}(s_p, s_i) \approx \Gamma[t]_r|_{p_1.....p_{i-1}.p})\theta_i\eta$. Therefore $\mathcal{M}_N \models (u_i|_p \approx \mathsf{app}(s_p, s_i))\theta_i\eta$ for every $p \in P$. Similarly, $\mathcal{M}_N \models (u' \approx u_i[\mathsf{app}(s_{q_i}, z)]_{q_i})\theta_i\eta$, hence:

$$\mathcal{M}_N \models (u' \approx u_i[\Gamma|_{p_1.....p_{i-1}.q_i}[\Delta[\Delta'[s]_{p_i.....p_n}]_{p_1.....p_{i-1}}]_{Q_{q_i}}]_{q_i})\theta_i\eta$$

and therefore $\mathcal{M}_N \models (u' \approx u_i[\Gamma|_{p_1.....p_{i-1}.q_i}[s]_{Q_{q_i}}]_{q_i})\theta_i\eta$, thus $\mathcal{M}_N \models (u' \approx z)\theta_i\eta$. We deduce that $\mathcal{M}_N \models (z \approx s_i)\theta_i\eta$, whence $\mathcal{M}_N \models (\Delta[z]_{p_1.....p_{i-1}} \approx \Delta[s_i]_{p_1.....p_{i-1}})\theta_i\eta$. Consequently, $\mathcal{M}_N \models s \approx t$.
- Otherwise, we must have $p_i = q_i$, for $i \in [1,n]$. Let $s'_{i+1} = u_i|_{p_i}$, for $i \in [1,n]$. The substitution $\theta_1 \ldots \theta_n$ is a solution of $s'_{n+1} \stackrel{?}{=} s_1 \wedge \bigwedge_{i=2}^{n} s'_i \stackrel{?}{=} s_i$, hence this problem admits an mgu σ, with $\theta_1 \ldots \theta_n = \sigma\eta$. It is easy to check that the sequence $(t_1, \ldots, t_{n+1}) = (s_1, \ldots, s_{n+1})\sigma$ is a cycle, thus there exists a k such that the sequence $(t_{1+k}, \ldots, t_{n+k}, t_{1+k})$ (with $t_i := t_{i-n}$ if $i > n$) is a cycle, and there exists a chain $(b_1, \ldots, b_{m+1})$ satisfying **Keep** such that $(t_{1+k}, \ldots, t_{n+k}, t_{1+k})$ is an extension of $(b_1, \ldots, b_{m+1})$. We assume that $(b_1, \ldots, b_{m+1})$ is the longest chain having this property, which entails that $(b_1, \ldots, b_{m+1})$ is eligible. The Uniq rule applies on the clauses $\mathcal{C}_{1+k}, \ldots, \mathcal{C}_{m+k}$, yielding a clause of the form $\mathcal{C}'_{1+k}\theta \vee \ldots \vee \mathcal{C}'_{m+k}\theta \vee \mathcal{E}$, where $\mathcal{E} = \big(\bigvee_{p \in P} u|_p \not\approx \mathsf{app}(s_p, b_1)\big) \vee u' \not\approx z \vee z \approx b_1$ and P is the set of positions p of some type $\tau \sim \text{type}(s_{1+k})$ in the constructor context $u = \Gamma_{1+k}[\ldots[\Gamma_{m+k}[\bullet]_{p_{m+k}} \ldots]_{p_{1+k}}]$ such that $p \not< p_{1+k}.....p_{m+k}$.

By Proposition 24, there exists a substitution σ' such that $\sigma = \theta\sigma'$ and $t_{i+k} = b_i\sigma'$ for $i \in [1, m+1]$. We have $\mathcal{C}'_i\theta\sigma'\eta = \mathcal{C}'_i\sigma\eta = \mathcal{C}_i\theta_i$, hence $\mathcal{M}_N \not\models \mathcal{C}'_{1+k}\theta\sigma'\eta \vee \ldots \vee \mathcal{C}'_{m+k}\theta\sigma'\eta$. Since N is saturated under Uniq, $\mathcal{M}_N \models \mathcal{C}'_{1+k}\theta \vee \ldots \vee \mathcal{C}'_{m+k}\theta \vee \mathcal{E}$, hence we deduce that $\mathcal{M}_N \models \mathcal{E}\sigma'\eta$.
It is clear that there exist constructor contexts Δ and $\Delta'[\bullet]_{r'}$ such that:

$$\Gamma[\Gamma[\bullet]_r]_r|_{p_1.....p_{1+k}} = \Delta[\Delta'[\bullet]_{r'}]_{p_{2+k}.....p_{n+k}}$$

By Proposition 8, for every position $p \in P$, there exists a term s'_p such that $\mathsf{App} \models \forall x.\, \mathsf{app}(s'_p, x) \approx \Gamma|_{p_1.....p_k.p}[\Delta[x]_{p_{2+k}.....p_{n+k}}]_{Q_p}$, where Q_p denotes the set of positions of $\bullet$ in $\Gamma|_{p_1.....p_k.p}$ (as in the previous case, this set contains at most one element). Let γ be a substitution mapping each variable s_p to s'_p and mapping z to $\Delta'[s]_{r'}$. As in the previous case, we observe that if $p \neq p_{1+k}.....p_{m+k}$, then $\Gamma|_{p_1.....p_k.p}$ contains no occurrences of $\bullet$, consequently:

$$s_p\gamma = \mathsf{cst}(\Gamma|_{p_1.....p_k.p}) = \mathsf{cst}(u_{1+k}[\ldots u_{m+k}[\bullet]_{p_{m+k}}]_{p_{1+k}}|_p\theta_{1+k}\ldots\theta_{m+k})$$

As in the previous case, it is easy to check that $\mathcal{M}_N \models (u|_p \approx \mathsf{app}(s_p, b_1))\sigma'\eta\gamma$ for every $p \in P$, and $\mathcal{M}_N \models (u' \approx z)\sigma'\eta\gamma$, thus we have $\mathcal{M}_N \models (z \approx b_1)\sigma'\eta\gamma$, whence $\mathcal{M}_N \models \Delta'[s]_{r'}\sigma'\eta \approx b_1\sigma'\eta$, thus:

$$\mathcal{M}_N \models \Gamma[\Delta[\Delta'[s]_{r'}\sigma'\eta]_{p_{2+k}.....p_{n+k}}]_{p_1.....p_{1+k}} \approx \Gamma[\Delta[b_1]_{p_{2+k}.....p_{n+k}}]_{p_1.....p_{1+k}}\sigma'\eta$$

and therefore $\mathcal{M}_N \models s \approx t$.

□

Definition 43. A signature is *coinductively nonbranching* if for every constructor $\mathsf{c} : \tau_1 \times \cdots \times \tau_n \to \tau$ such that $\tau \in \mathcal{T}_{\mathsf{coind}}$, there exists at most one $i \in [1, n]$ such that $\tau_i \sim \tau$.

For example, the signature is coinductively nonbranching for infinite streams and possibly infinite lists, but not for infinite binary trees.

Proposition 44. *Let t be a term of type $\tau \in \mathcal{T}_{\mathsf{coind}}$ and p_1, p_2 be two positions in t of types τ_1 and τ_2, respectively. If the signature is coinductively nonbranching, and if $\tau_1 \sim \tau_2 \sim \tau$, then $p_1 \leq p_2$ or $p_2 \leq p_1$.*

Proof. Assume that $p_i = p.j_i.p'_i$, where j_1 and j_2 are distinct integers. Then $t|_p$ is of the form $\mathsf{c}(t_1, \ldots, t_n)$, where c is a constructor, $j_1, j_2 \in [1, n]$, $\tau_i \rhd^* \mathrm{type}(t_{j_i})$ and $\mathrm{type}(t_{j_i}) \rhd^* \tau$ thus $\mathrm{type}(t_{j_1}) \sim \mathrm{type}(t_{j_2}) \sim \mathrm{type}(t|_p)$, with contradicts the fact that the signature is coinductively nonbranching. □

Corollary 45 (Fixpoints). *Assume that the signature is coinductively nonbranching. If $\mathsf{Cycl} \cup \mathsf{App} \subseteq N$ and $N \not\ni \square$ is saturated under $\mathcal{SP}$, Uniq, and Inf, then $\mathcal{M}_N$ satisfies condition* **FP**.

Proof. Let $\Gamma[\bullet]_P$ be a nonempty constructor context of type $\tau \in \mathcal{T}_{\mathsf{coind}}$, where P is a set of positions of type τ in Γ. By the hypothesis of the corollary, we may assume, by Proposition 44, that P is a singleton $\{r\}$. Since $\mathcal{M}_N \models \mathsf{Cycl}$, we have $\mathcal{M}_N \models \exists x.\, x \approx \Gamma[x]_r$. By Lemma 42, $\mathcal{M}_N \models x \approx \Gamma[x]_r \wedge y \approx \Gamma[y]_r \Rightarrow x \approx y$. □

Example 46. Corollary 45 does not hold for arbitrary signatures. The clause set $\{\mathsf{a} \approx \mathsf{c}(\mathsf{d}(\mathsf{a},\mathsf{b})), \mathsf{b} \approx \mathsf{e}(\mathsf{d}(\mathsf{a},\mathsf{b})), \mathsf{a}' \approx \mathsf{c}(\mathsf{d}(\mathsf{a}',\mathsf{b}')), \mathsf{b}' \approx \mathsf{e}(\mathsf{d}(\mathsf{a}',\mathsf{b}')), \mathsf{d}(\mathsf{a},\mathsf{b}) \not\approx \mathsf{d}(\mathsf{a}',\mathsf{b}')\}$ contradicts **FP**, because $\mathsf{d}(\mathsf{a},\mathsf{b})$ and $\mathsf{d}(\mathsf{a}',\mathsf{b}')$ are both solutions of $x \approx \mathsf{d}(\mathsf{c}(x),\mathsf{e}(x))$. However, the Uniq rule applies only with constructor contexts of head symbol c (if the chain starts with a or a') or e (if the chain starts with b or b').

Observe that in the proof of Lemma 42, the variables $p \in P$ (with the notations of the Uniq rule), are always instantiated with a term $\mathsf{cst}(u|_p)$, except when $p = q$. Thus the result holds for this particular instantiation of the rule, and all terms $\mathsf{app}(s_p, x)$ with $p \neq q$ may be replaced by $u|_p$ in this case. However, being able to instantiate s_p by terms different from $\mathsf{cst}(u|_p)$ is useful when contexts with several holes are considered, although the rule is not complete in this case. Note also that if the signature is coinductively nonbranching, then necessarily $P = \{q\}$, by Proposition 44.

6 Saturation Procedure

The inference rules of the calculus presented in Section 4 are all finitely branching, provided that the eligibility criterion is applied for the Acycl and Uniq rules. As a result, saturation of a clause set can be carried out using standard saturation procedures. These generally work by maintaining a set of *passive clauses* that initially contains all the clauses to saturate and a set of *active clauses* that is initially empty. The algorithm heuristically chooses a passive clause that becomes the *given clause*, moves it to the active clauses, and performs all possible inferences between it and the active clauses. Conclusions are added to the set of passive clauses, and the procedure is iterated until $\square$ is derived, or until the set of passive clauses is empty, in which case the set of active clauses is saturated.

To improve search, it is useful to distinguish between *simplifying rules* and *generating rules*. In simplifying rules, at least of one the premises is redundant with respect to the conclusion. The Inf rule is simplifying, as well as the Dist_1 and Inj_1 rules when the term t is not a variable, and the Acycl rule when there is only one premise and $t_1 = t_n$. Since they allow the replacement of a clause by another, simplifying rules should be applied immediately after the generation of a new clause. For the same reason, they should be applied to all literals (rather than only to maximal literals) and without any order condition.

In addition to the calculus, we propose the following simplifying rules to eliminate theory tautologies:

$$\frac{\mathsf{c}(\bar{s}) \not\approx \mathsf{d}(\bar{t}) \vee \mathcal{C}}{\varnothing}\ \mathsf{Dist}^- \qquad\qquad \frac{s \not\approx \Gamma[s] \vee \mathcal{C}}{\varnothing}\ \mathsf{Acycl}^-$$

where $\mathsf{c} \bowtie \mathsf{d}$, $\Gamma[\bullet]$ is a nonempty constructor context, and $\mathrm{type}(s) \in \mathcal{T}_{\mathsf{ind}}$. Moreover, the following rule applies injectivity of constructors $\mathsf{c} \in \mathcal{C}tr_{\mathsf{inj}}$ to simplify literals:

$$\frac{\mathsf{c}(s_1,\ldots,s_n) \not\approx \mathsf{c}(t_1,\ldots,t_n) \vee \mathcal{C}}{\left(\bigvee_{i=1}^{n} s_i \not\approx t_i\right) \vee \mathcal{C}}\ \mathsf{Inj}^-$$

The soundness of Inj^- follows from c's being a function symbol, but since it is also injective, the premise is redundant with respect to the theory. We conjecture that the addition of these simplification rules preserves refutational completeness.

If all constructors are free (i.e., $\mathcal{C}tr_{\mathsf{inj}} = \mathcal{C}tr$ and $\mathsf{c} \bowtie \mathsf{d}$ holds for all distinct constructors c and d), by applying the above rules eagerly, we also guarantee that in any literal $[\neg]s \approx t$ in an active clause, at most one of s or t has a constructor for head symbol, as (dis)equalities between constructor terms will have been simplified directly after clause generation. This invariant enables a few optimizations in the implementation of the generating rules, notably during the detection of chains.

The relaxation of the application conditions of the Sup rule increases the number of clauses it must generate and may hence be detrimental to the search. We can reduce the incidence of this scenario by choosing a term order that considers constructors as smaller than non-constructors. For path orders, we can choose a symbol precedence $\succ$ such that $\mathsf{f} \succ \mathsf{c}$ for all non-constructor symbols f and constructors c.

To implement the Acycl and Uniq rules, we must be able to efficiently detect eligible chains among the set of active clauses. Testing all subsets of the active clauses is impractical, and the detection of a chain requires the computation of an mgu over a set of equations, instead of a single equation. We present a procedure that takes the given clause $\mathcal{C}_1$ as input and applies the two rules to all subsets of clauses containing $\mathcal{C}_1$ and upon which an eligible chain can be built. There are three cases in which the rules must be applied: when the chain is a cycle, when it is variable-ended and has length 1, and when there exists an extension of the chain that violates **Keep**. The procedure relies on a data structure that provides a $\mathsf{nextLinks}(s')$ operation, where s' is a term. For each literal $s \approx t$ in an active clause $\mathcal{C}$ such that s is unifiable with s' under an mgu σ and $s\sigma \not\succeq t\sigma$, the operation returns the tuple $(\mathcal{C}, \sigma, T)$, where T is the set of terms under nonempty constructor positions in t. This operation can be implemented using term indexing techniques already found in state-of-the-art provers [21, Section 5.1].

The procedure $\mathsf{considerGiven}(\mathcal{C}_1)$ applies the rule Acycl or Uniq to all subsets of actives clauses that contain the given clause $\mathcal{C}_1$ and form an eligible chain:

Procedure $\mathsf{considerGiven}(\mathcal{C}_1)$ **is**
  for s_2' **such that** $\mathcal{C}_1 = s_1 \approx \Gamma[s_2'] \vee \mathcal{D}_1$ **do**
    $\mathsf{extendChain}(s_1, s_2', \{\}, \{\mathcal{C}_1\})$

Procedure $\mathsf{extendChain}(s_1, s_i', \theta, Ch)$ **is**
  if $s_1\theta = s_i'\theta$ **then**
    apply rule Acycl or Uniq to chain Ch under mgu θ
  else if s_i' is a variable **then**
    if $|Ch| = 1$ **then**
      apply rule Acycl or Uniq to chain Ch under mgu θ
  else if exists $(\mathcal{C}_i, \sigma, T) \in \mathsf{nextLinks}(s_i'\theta)$ **such that** $\mathcal{C}_i \in Ch$ **then**
    apply rule Acycl or Uniq to chain Ch under mgu θ
  else
    for $(\mathcal{C}_i, \sigma, T) \in \mathsf{nextLinks}(s_i'\theta)$ **do**
      $\mathsf{extendChain}(s_1, s_{i+1}', \sigma\theta, Ch \uplus \{\mathcal{C}_i\})$

7 Evaluation

We implemented the calculus presented above in the first-order theorem prover Vampire [12]. Our source code is publicly available.[1] The new rules are added to the existing calculus, which includes other sound rules and a sophisticated redundancy elimination mechanism. Vampire can process input files in SMT-LIB [4] format and recognizes both the `declare-datatypes` command and the nonstandard `declare-codatatypes` command. These commands trigger the addition of relevant axioms or the activation of inference rules, according to user-specified options. This implementation is an extension of previous work done in Vampire [11]. The behavior of this older implementation can be replicated by enabling only the simplification rules of the calculus and adding the axioms Dist, Inj, Exhaust, Sub, and NSub to the initial clause set.

We evaluated the implementation on 4170 problems that were used previously by Reynolds and Blanchette [19] to evaluate CVC4. These were generated by translating Isabelle problem to SMT-LIB using the Sledgehammer bridge [17] and on synthetic problems that exercise the properties of cyclic values. Both benchmark sets and detailed results are available online.[2]

All the experiments in this section were carried out on a cluster on which each node is equipped with two quad-core Intel processors running at 2.4 GHz, with 24 GB of memory. A 60 s time limit per problem was enforced. We used a single basic saturation strategy relying on the DISCOUNT saturation algorithm. The calculus was parameterized by a Knuth–Bendix term order, unless otherwise noted. This simple approach provides a homogeneous basis on which to compare the performance of the different parameters. It typically solves fewer problems than the portfolio approach commonly used with Vampire and other provers, in which several different strategies are tried in short time slices.

We first compare the performance of three configurations of the prover on the Isabelle problems. The first configuration corresponds to the axiomatic approach presented in Section 3: the axioms Dist, Inj, Exhaust, Sub, NSub, App, Uniq, Cycl, and Hole are added to the set of clauses to saturate, and only standard inferences rules are used by the prover. Superposition need not rewrite the nonmaximal side of an equation.

The second configuration implements part of the calculus presented in Section 4. Only the axioms Exhaust, Sub, NSub, App, Uniq, Cycl, and Hole are added to the clauses, and the rules Dist_1, Dist_2, Inj_1, and Inj_2 are used during the search, in addition to the simplification rules described in Section 6. The side conditions of Sup are also relaxed. The rules Acycl and Uniq are not used; instead, reasoning on the properties of cyclic terms is based on axioms.

The third configuration uses all the rules described in Section 4. Only the axioms Sub and App are added, on which the Acycl and Uniq rules depend, and the axioms Cycl and Exhaust. This configuration is the only one which does not ensure refutational completeness, since Uniq is incomplete with respect to the uniqueness of fixpoints for branching codatatypes.

[1] `http://github.com/vprover/vampire/releases/tag/ijcar2018-data`

[2] `http://matryoshka.gforge.inria.fr/pubs/supdata_data.tar.gz`

The first two configurations both solved 1114 problems and the third one solved 1113 problems; 1116 problems are solved by at least one configuration. These homogeneous results do not reveal significant differences between the approaches. To assess the role of the acyclicity property of datatypes and the properties of codatatype fixpoints in the benchmarks, we also tested a system that did not include any axioms and rules related to these properties. With such an incomplete system, we found that 12 problems could not be solved. This is roughly in line with the results of Reynolds and Blanchette using CVC4 on the same problems [19]. No new problems were solved by this configuration, suggesting that reasoning about properties of cyclic terms does not lead to worse performance even when these properties are not needed for refutation.

We also tested variants of the last two configurations in which the calculus was parameterized by a lexicographic path order, to assess whether this term order could improve the performance when used with the relaxed superposition rule. These configurations solved a total of 1104 problems, including 5 new problems. This shows that using a different term order allows the exploration of different parts of the search space, but the choice of order does not seem to impact the performance of the relaxed superposition rule.

Since properties of cyclic values are seldom used in the Isabelle benchmarks, we crafted a set of (refutable) problems to assess the performance of the rules Acycl and Uniq. For a term s and a nonconstant context $\Gamma[\bullet]$, let $\mathsf{exchain}(s,\Gamma[\bullet])$ denote any sentence $\exists s_2,\ldots s_n \forall t_1,\ldots t_m.\ s \approx \Gamma_1[s_2] \wedge \cdots \wedge s_n \approx \Gamma_n[s]$, where $t_1,\ldots,t_m$ all occur in Γ and such that $\Gamma_1[\ldots[\Gamma_n[\bullet]]\ldots] = \Gamma[\bullet]$. The formula $\exists s.\, \mathsf{exchain}(s,\Gamma[\bullet])$, where $\mathrm{type}(s) \in \mathcal{T}_{\mathsf{ind}}$, forms an acyclicity problem. The set of acyclicity problems used in our experiments is denoted AC. If $m = 0$, the clausified form of this problem is ground (ACG). The formula $\exists s_1, s_2.\, \mathsf{exchain}(s_1,\Gamma[\bullet]) \wedge \mathsf{exchain}(s_2,\Gamma[\bullet]) \wedge s_1 \not\approx s_2$, where $\mathrm{type}(s_1) \in \mathcal{T}_{\mathsf{coind}}$, forms a uniqueness problem (U). Note that in such a problem, the two chains may not be formed upon the same equalities, although they build the same constructor context. Similarly, if $m = 0$, we obtain a ground uniqueness problem (UG). Finally, the sentence $\forall s.\ \neg\, \mathsf{exchain}(s,\Gamma[\bullet])$, for $\mathrm{type}(s) \in \mathcal{T}_{\mathsf{coind}}$, forms an existence problem (EX).

We generated 100 instances of each type of problem. The number of problems solved by Vampire (V) on these problems are presented in the following table, along with the results obtained using CVC4's [3] and Z3's [14] native support for datatypes and, in CVC4's case, for codatatypes:

	AC			ACG			U		UG		EX	
	V	CVC4	Z3	V	CVC4	Z3	V	CVC4	V	CVC4	V	CVC4
Axioms	65	–	–	**100**	–	–	**14**	–	10	–	**40**	–
Calculus	82	**100**	59	**100**	**100**	**100**	**14**	12	13	**100**	35	0

The number of problems solved shows that the Acycl rule performs better than the axioms for acyclicity problems with variables. Only one of these problems could be solved by the axiomatic approach and not by the Acycl rule. Both approaches managed to solve all of the ground acyclicity problems. Z3 solved all of the ground problems, performing slightly less well on those featuring universal quantifiers. CVC4 was able to solve all of the acyclicity problems, including those with universal quantifiers, a notable improvement over previous results obtained on similar problems [21, Section 6].

On uniqueness problems, the Uniq rule solved a superset of the ground problems solved by the axiomatic approach, whereas on nonground problems each approach uniquely solved 3 problems, for a total of 17 problems solved. Again, CVC4 performed remarkably well on ground problems, while the presence of variables in the problem led to a marked degradation of its performance. Finally, for existence problems, the refutation relies mostly on the Cycl axiom, which is included in the clause set in both Vampire configurations. Yet, the purely axiomatic approach was able to solve 6 problems that could not be solved when the Uniq rule was activated, indicating that the rule might lead the search in a suboptimal direction. The theory solver in CVC4 does not take into account the existence of fixpoints for codatatypes, which is a nonground property. Consequently, none of the existence problems were solved by CVC4.

From the results, it would appear that the calculus supersedes the axiomatic approach for problems with datatypes. For codatatypes, both approaches are able to solve different problems, suggesting that they should both be included in a strategy portfolio.

8 Related Work

The potential of (co)datatypes for automated reasoning has been studied mostly in the context of satisfiability modulo theories (SMT). Datatypes are parts of the SMT-LIB 2.6 standard [4]. They were implemented in CVC3 by Barrett et al. [5], in Z3 [14] by de Moura, and in CVC4 by Reynolds and Blanchette [19]. The CVC4 work also includes a decision procedure for the ground theory of codatatypes. Moreover, CVC4 supports automatic structural induction [20] and dedicated reasoning support for selectors.

Structural induction has also been added to superposition by Kersani and Peltier [10], Cruanes [9], and Wand [23]. In unpublished work, Wand implemented incomplete inference rules for datatypes, including acyclicity, in his superposition prover Pirate. Robillard's earlier Acycl rule [21] has inspired our Acycl rule, but it suffered from many forms of incompleteness. For example, given the unsatisfiable clause set $\{\mathsf{a} \approx \mathsf{c}(x) \vee \mathsf{p}(x), \neg\mathsf{p}(\mathsf{c}(\mathsf{a}))\}$, the old Acycl rule derived only $\mathsf{p}(\mathsf{a})$ before reaching saturation. Another issue concerned cycles built from multiple copies of the same premise. Consider the unsatisfiable clause set $\{\mathsf{a} \approx \mathsf{c}(\mathsf{f}(\mathsf{zero})), \mathsf{f}(x) \approx \mathsf{c}(\mathsf{f}(\mathsf{suc}(x))), \mathsf{f}(\mathsf{suc}(\mathsf{suc}(\mathsf{zero}))) \approx \mathsf{c}(\mathsf{a})\}$. The new rule can build a cycle of length 5 by using the second clause twice, with $x = \mathsf{zero}$ and $x = \mathsf{suc}(\mathsf{zero})$, whereas the old rule never reused clauses, to achieve finite branching. Our solution to achieve finite branching involves using the sub predicate, pushing the burden of enumerating possibly infinitely many cycles onto the core superposition calculus.

In the context of program verification, Bjørner [6] introduced a decision procedure for (co)datatypes in STeP, the Stanford Temporal Prover. The program verification tool Dafny provides both a syntax for defining (co)datatypes and some support for automatic (co)induction proofs [13]. Other verification tools such as Leon [22] and RADA [18] also include (semi-)decision procedures for datatypes. We refer to Barrett et al. [5] and Reynolds and Blanchette [19] for further discussions of related work.

9 Conclusion

We presented two approaches to reason about datatypes and codatatypes in first-order logic: an axiomatization and an extension of the superposition calculus. We established completeness results about both. We also showed how to integrate the new inference rules in a saturation prover's main loop and implemented them in the Vampire prover. The empirical results look promising, although it is not clear from our benchmarks how often the most difficult properties—acyclicity for datatypes, existence and uniqueness of fixpoints for codatatypes—are useful in practice.

This work is part of a wider research program that aims at bridging the gap between automatic theorem provers and their applications to program verification and interactive theorem proving. In future work, we want to reconstruct the new proof rules in Isabelle, to make it possible to enable datatype reasoning in Sledgehammer. We also believe that further tuning and evaluations could help improve the calculus and the heuristics.

Acknowledgment. We thank Alexander Bentkamp, Simon Cruanes, Uwe Waldmann, Daniel Wand, and Christoph Weidenbach for fruitful discussions that led to this work. We also thank Mark Summerfield and the anonymous reviewers for suggesting textual improvements.

Blanchette has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). Robillard has received funding from the ERC Starting Grant 2014 SYMCAR 639270, the Wallenberg Academy Fellowship 2014 TheProSE, the Swedish Research Council grant GenPro D0497701, and the Austrian FWF research project RiSE S11409-N23.

References

[1] Bachmair, L., Dershowitz, N., Hsiang, J.: Orderings for equational proofs. In: LICS '86. pp. 346–357. IEEE Computer Society (1986)

[2] Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. 4(3), 217–247 (1994)

[3] Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer (2011)

[4] Barrett, C., Fontaine, P., Tinelli, C.: The SMT-LIB standard: Version 2.7. Tech. rep., University of Iowa (2017), `http://smt-lib.org/`

[5] Barrett, C., Shikanian, I., Tinelli, C.: An abstract decision procedure for satisfiability in the theory of inductive data types. J. Satisf. Boolean Model. Comput. 3, 21–46 (2007)

[6] Bjørner, N.S.: Integrating Decision Procedures for Temporal Verification. Ph.D. thesis, Stanford University (1998)

[7] Blanchette, J.C., Hölzl, J., Lochbihler, A., Panny, L., Popescu, A., Traytel, D.: Truly modular (co)datatypes for Isabelle/HOL. In: Klein, G., Gamboa, R. (eds.) ITP 2014. LNCS, vol. 8558, pp. 93–110. Springer (2014)

[8] Comon, H., Lescanne, P.: Equational problems and disunification. J. Symb. Comput 7(3–4), 371–425 (1989)

[9] Cruanes, S.: Superposition with structural induction. In: Dixon, C., Finger, M. (eds.) FroCoS 2017. LNCS, vol. 10483, pp. 172–188. Springer (2017)

[10] Kersani, A., Peltier, N.: Combining superposition and induction: A practical realization. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) FroCoS 2013. LNCS, vol. 8152, pp. 7–22. Springer (2013)
[11] Kovács, L., Robillard, S., Voronkov, A.: Coming to terms with quantified reasoning. In: Castagna, G., Gordon, A.D. (eds.) POPL 2017. pp. 260–270. ACM (2017)
[12] Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification (CAV 2013). LNCS, vol. 8044, pp. 1–35. Springer (2013)
[13] Leino, K.R.M., Moskal, M.: Co-induction simply—Automatic co-inductive proofs in a program verifier. In: Jones, C.B., Pihlajasaari, P., Sun, J. (eds.) FM 2014. LNCS, vol. 8442, pp. 382–398. Springer (2014)
[14] de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer (2008)
[15] Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 371–443. Elsevier and MIT Press (2001)
[16] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
[17] Paulson, L.C., Blanchette, J.C.: Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In: Sutcliffe, G., Schulz, S., Ternovska, E. (eds.) IWIL-2010. EPiC, vol. 2, pp. 1–11. EasyChair (2012)
[18] Pham, T., Whalen, M.W.: RADA: A tool for reasoning about algebraic data types with abstractions. In: Meyer, B., Baresi, L., Mezini, M. (eds.) ESEC/FSE '13. pp. 611–614. ACM (2013)
[19] Reynolds, A., Blanchette, J.C.: A decision procedure for (co)datatypes in SMT solvers. J. Autom. Reason. 58(3), 341–362 (2017)
[20] Reynolds, A., Kuncak, V.: Induction for SMT solvers. In: D'Souza, D., Lal, A., Larsen, K.G. (eds.) VMCAI 2015. LNCS, vol. 8931, pp. 80–98. Springer (2014)
[21] Robillard, S.: An inference rule for the acyclicity property of term algebras. In: Kovács, L., Voronkov, A. (eds.) Vampire 2017. EPiC, EasyChair, to appear
[22] Suter, P., Köksal, A.S., Kuncak, V.: Satisfiability modulo recursive programs. In: Yahav, E. (ed.) SAS 2011. LNCS, vol. 6887, pp. 298–315. Springer (2011)
[23] Wand, D.: Superposition: Types and Polymorphism. Ph.D. thesis, Universität des Saarlandes (2017)