A Complete Cyclic Proof System for Inductive Entailments in First Order Logic

Radu Iosif, Cristina Serban

A Complete Cyclic Proof System for Inductive Entailments in First Order Logic

Radu Iosif[1] and Cristina Serban[1]

Univ. Grenoble Alpes, CNRS, VERIMAG, F-38000 Grenoble France

Abstract

In this paper we develop a cyclic proof system for the problem of inclusion between the least sets of models of mutually recursive predicates, when the ground constraints in the inductive definitions are quantifier-free formulae of first order logic. The proof system consists of a small set of inference rules, inspired by a top-down language inclusion algorithm for tree automata [9]. We show the proof system to be sound, in general, and complete, under certain semantic restrictions involving the set of constraints in the inductive system. Moreover, we investigate the computational complexity of checking these restrictions, when the function symbols in the logic are given the canonical Herbrand interpretation.

1 Introduction

Inductive definitions play an important role in computing, being an essential component of programming languages, databases, automated reasoning and program verification systems. The main advantage of using inductive definitions is the ability of reasoning about sets of logical objects, by means of recursion. The semantics of these definitions is described in terms of least fixed points of higher-order functions on assignments mapping predicates to sets of models. A natural problem is the *entailment*, that asks whether the least solution of one predicate is included in the least solution of another. Examples of entailments are language inclusion between finite-state (tree) automata, context-free grammars or verification conditions generated by shape analysis tools using specifications of recursive data structures as contracts of program correctness.

The principle of *infinite descent*, stated by Fermat [5], has become an important tool for reasoning about entailments between inductively defined predicates. In a nutshell, a proof by infinite descent is a particular proof by contradiction, in which we assume the existence of a counterexample from a well-founded domain and show the existence of a strictly smaller counterexample for the same entailment problem. By repeating this step, we obtain an infinite descending chain of counterexamples, which is impossible when the domain of counterexamples is well-founded.

A *cyclic proof* [3] is a finite derivation tree in which certain leaves (buds) are connected to internal nodes (companions) via backlinks, such that on every infinite path obtained by following backlinks there are infinitely many *progress points* ensuring that no counterexample can be propagated along that path *ad infinitum*. Together with the local soundness of each proof rule (if all premises are true, the conclusion is true), this so-called *global trace condition* [2, 4] ensures the soundness of a cyclic proof.

In particular, cyclic proofs reason about inductive definitions by deriving the inductive invariants on-the-fly, as opposed to classical proofs by induction, in which these invariants have to be provided. Moreover, cyclic proofs are shown to be strictly more general than inductive proofs: all proofs by induction can be written as cyclic proofs (with cut) [4], whereas there are statements provable using cyclic proof that cannot be proved using induction [1]. Despite these interesting features, cyclic proofs face several problems, e.g.:

- Practical applications within automated reasoning systems are met with the inherent complexity of checking the global trace condition, which requires time exponential in the size of the derivation

tree, unless certain structural restrictions are used. A typical restriction is that the companion of a bud must also be an ancestor, which can be checked in time logarithmic in the size of the proof. Additionally, it is possible to check the soundness of a cyclic proof by checking inclusion of weighted automata [14].

- The boundaries of completeness (for which systems of inductive definitions does a valid entailment query have a proof?) are not entirely understood. In [3, 4], cut-free completeness is shown for the system LKID$^{\omega}$, in which one allows infinite recursive proofs, that cannot be folded into finite cyclic proofs. It is thus an interesting open problem which inductive systems admit complete (finite) cyclic proofs. Besides implying the decidability of entailments[1], a complete proof system gives a way of *certifying* the correctness of a certain theorem prover implementation, by producing machine-checkable proofs of the validity of queries [17].

The relation between cyclic proof systems and automata is long-standing, one of the main results being a complete proof system for the modal μ-calculus, proposed by Walukiewicz [18]. In a similar vein, the proof system presented in this paper stems from the observation that language inclusion of tree automata can be proved using an infinite descent argument. Given tree automata A and B, we assume the existence of a tree $t \in \mathcal{L}(A) \setminus \mathcal{L}(B)$, where $\mathcal{L}(A)$ [$\mathcal{L}(B)$] denotes the language of A [B], and prove the existence of strictly smaller tree $t' \in \mathcal{L}(A) \setminus \mathcal{L}(B)$, in the subtree order. Since the subtree order is well-founded, there is no such tree to start with and thus $\mathcal{L}(A) \subseteq \mathcal{L}(B)$. Moreover, the search space built by antichain-based algorithms for language inclusion between tree automata [9] can roughly be viewed as a cyclic sequent calculus proof. Because tree automata over finite alphabets have a decidable inclusion problem, these problems are further shown to have complete cyclic proof systems.

The contributions of this paper are two-fold:

1. We provide a proof system for entailments between inductive predicates defined using a system of mutually recursive definitions, written using quantifier-free first-order logic. We show that the system is sound, using a local constraint on the (reversed) path between a bud and its companion in the derivation. Completeness is subject to three decidable conditions on the set of ground constraints that occur in the system of definitions. Moreover, we make proof generation effective by providing suitable strategies that limit the possibilities of applying the inference rules and guide the search towards finding a proof or a counterexample.
2. We investigate the computational complexity of checking whether a given system complies with the semantic restrictions required to ensure its completeness. An implementation can effectively check these conditions and issue a warning indicating the source of incompleteness of the proof system, for a given entailment problem.

Our proof system is a typical example of the *unfold-and-match* method that is used in state-of-the-art tools for reasoning about inductively defined recursive data structures [11]. Our main contribution in this area is the study of the completeness problem and an attempt to defining a class of inductive definitions for which unfold-and-match paradigms are complete, when infinite descent is used to ensure termination. As a by-product, we obtain a language inclusion algorithm for tree automata, that in addition to answering yes/no, issues proof certificates and counterexamples.

The method presented here is general enough so that it can be adapted to other types of logic. We provide a prototype implementation of this semi-algorithm [10], that uses a slightly modified variant of the inference rules for proving entailments between inductive predicates in Separation Logic [13].

[1] Assuming a countable vocabulary and a complete finite proof system, since any entailment has either a countermodel or a proof, one could decide its validity by alternating the search for a countermodel with that of a proof.

2 Preliminaries

We adopt the classical terminology of first-order logic over a countable signature Σ consisting of function symbols $f(x_1,\ldots,x_n)$, where $\#(f)=n$ denotes the arity of f. A constant c is a function symbol of arity zero. We assume the existence of a boolean sort with constants $\bot$ and $\top$ for false and true, respectively. A *predicate* $p(x_1,\ldots,x_n)$ is a function symbol of the boolean sort.

Let Var be a countably infinite set of variables. Terms are defined recursively: any constant or variable is a term and $f(t_1,\ldots,t_{\#(f)})$ is a term if $t_1,\ldots,t_{\#(f)}$ are terms. We write $\mathcal{T}_\Sigma(\mathbf{x})$ for the set of terms with function symbols in Σ and variables in $\mathbf{x}$. A *ground term* is a term without variables and $\mathcal{T}_\Sigma$ denotes the set of ground terms (we drop the subscript when it is clear from the context). An *atom* is either an equality $t_1 \approx t_2$, or a predicate atom $p(t_1,\ldots,t_{\#(p)})$, where $t_1,t_2,\ldots,t_{\#(p)}$ are terms. A *formula* is a quantified boolean combination of atoms. The *size* of a formula is the number of symbols needed to write it down.

For a term t, we denote by $\mathrm{FV}(t)$ the set of variables in t. For a formula ϕ, we denote by $\mathrm{FV}(\phi)$ the set of variables not occurring under the scope of a quantifier and write $\phi(\mathbf{x})$ for $\mathrm{FV}(\phi)\subseteq\mathbf{x}$. The same notation applies to sets F of formulae, where $\mathrm{FV}(F)\stackrel{\mathsf{def}}{=}\bigcup_{\phi\in F}\mathrm{FV}(\phi)$.

Given sets of variables $\mathbf{x}$ and $\mathbf{y}$, a *substitution* $\theta:\mathbf{x}\to\mathcal{T}_\Sigma(\mathbf{y})$ is a function mapping variables to terms and let $\mathbf{x}\theta\stackrel{\mathsf{def}}{=}\{\theta(x)\mid x\in\mathbf{x}\}$ be the image of $\mathbf{x}$ via θ. A substitution θ is *flat* if $\mathsf{Var}\theta\subseteq\mathsf{Var}$, i.e. each variable is mapped to a variable. For a formula $\phi(\mathbf{x})$, we denote by $\phi\theta$ the formula obtained by replacing each occurrence of $x\in\mathbf{x}$ with the term $\theta(x)$, and lift this notation to sets as $F\theta\stackrel{\mathsf{def}}{=}\{\phi\theta\mid\phi\in F\}$.

A *structure* is a tuple (U,I,ν), where U is a *universe*, I is an *interpretation* that maps each function symbol f into a total function $f^I:\mathsf{U}^{\#(f)}\to\mathsf{U}$ and ν is a partial mapping of variables into elements of U. For a tuple of variables $\mathbf{x}=(x_1,\ldots,x_n)$, we write $\nu(\mathbf{x})$ for $(\nu(x_1),\ldots,\nu(x_n))$. A distinguished class of structures are the *Herbrand* structures $(\mathcal{T}_\Sigma,\mathcal{H},\nu)$, where $\mathcal{H}$ maps each function symbol f into the function $f^{\mathcal{H}}$ carrying any tuple of terms $t_1,\ldots,t_{\#(f)}$ into the term $f(t_1,\ldots,t_{\#(f)})$.

The semantics of first-order logic is given by the $\models$ relation between structures and formulae. For a quantifier-free formula ϕ, we write $(\mathsf{U},I,\nu)\models\phi$ when the formula obtained from ϕ by replacing each function symbol f by f^I and each variable x by $\nu(x)$ is logically equivalent to $\top$. The interpretation of quantifiers is standard: $(\mathsf{U},I,\nu)\models\exists x\,.\,\phi$ if and only if $(\mathsf{U},I,\nu[x\leftarrow u])\models\phi$ for some value $u\in\mathsf{U}$, where $\nu[x\leftarrow u]$ is the valuation that assigns x to u and behaves like ν elsewhere.

When U and I are understood, we write $\nu\models\phi$ instead of $(\mathsf{U},I,\nu)\models\phi$. A formula ϕ is *satisfiable* if and only if $(\mathsf{U},I,\nu)\models\phi$ for some structure, called a *model*. We say that ϕ *entails* ψ, written $\phi\models\psi$ if and only if every model of ϕ is also a model of ψ.

2.1 Inductive Definitions

Let $\mathsf{Pred}\subseteq\Sigma$ be a finite set of predicates, whose interpretations are given by a set of rules. Formally, a *rule* is a pair $(\{\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n),q_1(\mathbf{x}_1),\ldots,q_n(\mathbf{x}_n)\},p(\mathbf{x}))$, where $\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n$ are pairwise disjoint sets of variables, ϕ is a formula, called the *constraint*, $p(\mathbf{x})$ is a predicate atom called the *goal* and $q_1(\mathbf{x}_1),\ldots,q_n(\mathbf{x}_n)$ are predicate atoms called *subgoals*. The variables in $\mathbf{x}$ are called *goal variables*, whereas those in $\bigcup_{i=1}^n\mathbf{x}_i$ are called *subgoal variables*.

A *system of inductive definitions* (system, for short) S is a finite set of rules, such that for each $\rho\in\mathsf{S}$, the constraint of ρ is a satisfiable conjunctive quantifier-free formula[2]. W.l.o.g, we assume that each goal with a predicate symbol p has the same tuple goal variables and that each predicate symbol $p\in\mathsf{Pred}$ occurs within the goal of at least one rule in S. As a shorthand, we write $p(\mathbf{x})\coloneqq_{\mathsf{S}}R_1\mid\ldots\mid R_n$, when $(R_1,p(\mathbf{x})),\ldots,(R_n,p(\mathbf{x}))$ are all the rules with goal $p(\mathbf{x})$ in S.

[2] A rule with a false constraint is removed and $(\{\phi_1\vee\ldots\vee\phi_n,Q\},p(x))$ is replaced by $(\{\phi_1,Q\},p(x)),\ldots,(\{\phi_n,Q\},p(x))$.

Given a universe U and an interpretation I, an *assignment* $\mathcal{X}$ maps each predicate $p \in \mathsf{Pred}$ to a set $\mathcal{X}(p) \subseteq \mathsf{U}^{\#(p)}$. Given a rule $(R, p(\mathbf{x})) \in \mathsf{S}$, where $R = \{\phi, q_1(\mathbf{x}_1), \ldots, q_n(\mathbf{x}_n)\}$, we extend the assignment to the body of the rule as $\mathcal{X}(R) \stackrel{\text{def}}{=} \{\nu \mid (\mathsf{U}, I, \nu) \models \phi \text{ and } \forall i \in [1,m] \,.\, \nu(\mathbf{x}_i) \in \mathcal{X}(q_i)\}$. Then we define the function $\mathbb{F}_\mathsf{S}$ on assignments, such that, for each assignment $\mathcal{X}$ and each predicate $p \in \mathsf{Pred}$, we have:

$$(\mathbb{F}_\mathsf{S}(\mathcal{X}))(p) = \bigcup_{i=1}^{n} \{\nu(\mathbf{x}) \mid \nu \in \mathcal{X}(R_i)\}, \text{ where } p(\mathbf{x}) :=_\mathsf{S} R_1 \mid \ldots \mid R_n \ .$$

A *solution* of S is an assignment $\mathcal{X}$ such that $\mathbb{F}_\mathsf{S}(\mathcal{X}) \subseteq \mathcal{X}$, where inclusion between assignments is defined pointwise. It can easily be shown that the set of assignments, together with the $\subseteq$ relation, forms a complete lattice, since any powerset equipped with the subset relation is a complete lattice. Because $\mathbb{F}_\mathsf{S}$ is monotone[3], the assignment $\mu\mathsf{S} \stackrel{\text{def}}{=} \bigcap\{\mathcal{X} \mid \mathbb{F}_\mathsf{S}(\mathcal{X}) \subseteq \mathcal{X}\}$ is the least fixpoint of $\mathbb{F}_\mathsf{S}$ and the least solution of S, where intersection of assignments is again defined pointwise.

Example 1. *Consider the system S_{neo} [3, example 2.2.5 and 2.2.6] consisting of the following rules, where c is a constant and s is a monadic function symbol:*

$$n(x) :=_{\mathsf{S}_{neo}} x \approx c \mid x \approx s(y), n(y) \qquad e(x) :=_{\mathsf{S}_{neo}} x \approx c \mid x \approx s(y), o(y) \qquad o(x) :=_{\mathsf{S}_{neo}} x \approx s(y), e(y)$$

Under the Herbrand interpretation, the least solution is $\mu\mathsf{S}_{neo}(n) = \{s^i(c) \mid i \geq 0\}$, $\mu\mathsf{S}_{neo}(e) = \{s^{2i}(c) \mid i \geq 0\}$ and $\mu\mathsf{S}_{neo}(o) = \{s^{2i+1}(c) \mid i \geq 0\}$, i.e. n, e and o are sets of terms corresponding to the unary encodings of natural, even and odd numbers, respectively. ■

Furthermore, the $\mathbb{F}_\mathsf{S}$ functions are provably continuous[4], thus $\mu\mathsf{S} = \bigcup_{i \geq 1} \mathbb{F}^i_\mathsf{S}(\emptyset)$, where $\emptyset$ is the assignment mapping each predicate to the empty set and $\mathbb{F}^i_\mathsf{S}$ is the i-th iteration of $\mathbb{F}_\mathsf{S}$, i.e. $\mathbb{F}^1_\mathsf{S} \stackrel{\text{def}}{=} \mathbb{F}_\mathsf{S}$ and $\mathbb{F}^i_\mathsf{S} \stackrel{\text{def}}{=} \mathbb{F}^{i-1}_\mathsf{S} \circ \mathbb{F}_\mathsf{S}$, for all $i > 1$. Continuity of $\mathbb{F}_\mathsf{S}$ allows us to compute the sequence of under-approximants of $\mu\mathsf{S}$ by iterating over natural numbers, instead of ordinals. For a predicate p and a tuple of values $\mathbf{v} = (v_1, \ldots, v_{\#(p)}) \in \mathsf{U}^{\#(p)}$, we denote by $sn(p, \mathbf{v}) \in \mathbb{N}$ the least number $j > 0$ such that $\mathbf{v} \in \bigcup_{i=1}^{j} \mathbb{F}^i_\mathsf{S}(\emptyset)(p)$, or 0 if there is no such j. This value is called a *stage number* in the following.

In this paper, we are concerned with the following *entailment problem*: given an inductive system S and predicates $p, q_1, \ldots, q_n$ having the same arity, is it true that $\mu\mathsf{S}(p) \subseteq \bigcup_{i=1}^{n} \mu\mathsf{S}(q_i)$? We denote valid entailments by $p \models_\mathsf{S} q_1, \ldots, q_n$. For instance, considering the system S_{neo} from Example 1, we have $n \models_{\mathsf{S}_{neo}} e, o$, whereas $n \not\models_{\mathsf{S}_{neo}} e$ and $n \not\models_{\mathsf{S}_{neo}} o$.

We close this section by showing that the entailment problem is undecidable, in particular, when restricted to Herbrand structures, with universe $\mathcal{T}_\Sigma$ and interpretation of function symbols $\mathcal{H}$.

Theorem 1. *The entailment problem on Herbrand structures is undecidable.*

Proof. The proof is by reduction from the inclusion problem for context-free languages, a known undecidable problem [16, Theorem 5.10]. Let $G = \langle \Xi, \Sigma, \Delta \rangle$ be a context-free grammar, where Ξ is the set of nonterminals, Σ is the alphabet of terminals, and Δ is a set of productions $(X, w) \in \Xi \times (\Xi \cup \Sigma)^*$. For a nonterminal $X \in \Xi$, we denote by $L_X(G) \subseteq \Sigma^*$ the language produced by G starting with X as axiom. The problem "given $X, Y \in \Xi$, does $L_X(G) \subseteq L_Y(G)$?" is undecidable. Given a context-free grammar $G = \langle \Xi, \Sigma, \Delta \rangle$, we define a system S_G as follows:

- each nonterminal $X \in \Xi$ corresponds to a predicate $X(x, y)$,
- each alphabet symbol $a \in \Sigma$ corresponds to a unary function symbol $\overline{a}$ and a word $w = a_1 \ldots a_n \in \Sigma^*$ is encoded by the context (i.e. the term with a hole) $\overline{w} = \overline{a_1}(\ldots \overline{a_n}(.))$,

[3] See [15, Theorem 1.3.2].
[4] See [15, Theorem 1.3.4].

- each grammar rule $(X, u_1X_1 \ldots u_nX_nu_{n+1}) \in \Delta$ corresponds to a rule:

$$\langle \{\phi(x,y,x_1,y_1,\ldots,x_n,y_n), X_1(x_1,y_1), \ldots, X_n(x_n,y_n)\}, X(x,y)\rangle$$

of S_G, where $\phi \equiv x \approx \overline{u_1}(x_1) \ \wedge \ \bigwedge_{i=1}^{n-1} y_i \approx \overline{u_{i+1}}(x_{i+1}) \ \wedge \ y_n \approx \overline{u_{n+1}}(y)$. In particular, a rule $(\varepsilon, X) \in \Delta$ maps into a rule $\langle \{x \approx y\}, X(x,y)\rangle$ of S_G.

We must check that, for any nonterminals $X, Y \in \Xi$, we have $L_X(G) \subseteq L_Y(G)$ if and only if $X \models_{\mathsf{S}_G} Y$. This is proved using the following invariant:

$$\forall w \in \Sigma^* \ . \ (\forall t \ . \ [x \leftarrow \overline{w}(t), y \leftarrow t] \in \mu\mathsf{S}_G(X)) \Leftrightarrow w \in L_X(G)$$

where $[x \leftarrow t, y \leftarrow u]$ denotes the valuation mapping x to t and y to u. □

3 Cyclic Proofs for Inductive Predicate Entailments

The previous negative result (Theorem 1) motivates the search for classes of inductive definitions with decidable entailment problems. Since inductive definitions that encode context-free grammars cannot have decidable entailments, an obvious workaround is to consider systems that define regular languages. In this respect, trees are arguably the most general type of structure for which the notion of regularity (recognizability by finite-state automata) is well understood[5].

For this reason, the developments in this section are inspired by a state-of-the-art language inclusion algorithm for top-down tree automata [9], briefly explained next. Our method for developing a proof system (set of inference rule schemata) for the entailment problem is to losely mimic the actions of the algorithm and the order in which they apply. However, we go beyond just proving inclusion of tree automata. Our proof system is sound and can be used with any entailment problem as defined above. Moreover, we characterize the class of inductive systems for which valid entailments do have finite proofs built using a strategy we provide.

3.1 Tree Automata Inclusion as Cyclic Proof Search

We assume basic knowledge of tree automata [6] and consider top-down nondeterministic finite tree automata (NFTA), whose actions are described by transition rules $q \xrightarrow{f} (q_1, \ldots, q_n)$, meaning that if the automaton is in state q and the input is a ground term $f(t_1, \ldots, t_n)$, then it moves simultaneously on each t_i changing its state to q_i, for all $i \in [n]$. A ground term is *accepted* by an automaton A in state q if each constant subterm (leaf) can be eventually read by a rule of the form $q \xrightarrow{a} ()$. The language of a state q in A, denoted $\mathcal{L}(A,q)$, is the set of ground terms accepted by A starting with q.

An NFTA can be naturally viewed as an inductive system, where predicates represent states and predicate rules are obtained directly from transition rules, as follows. For instance, the transition rule $q \xrightarrow{f} (q_1, \ldots, q_n)$ can be written as $(\{x \approx f(x_1, \ldots, x_n), q_1(x_1), \ldots, q_n(x_n)\}, q(x))$, where variables range over ground terms and the function symbols are interpreted in the Herbrand sense. If S_A is the inductive system corresponding to A, then $\mathcal{L}(A,q) = \mu\mathsf{S}_A(q)$ for any state (predicate) q and the language inclusion problem $\mathcal{L}(A,p) \subseteq \bigcup_{i=1}^{n} \mathcal{L}(A,q_i)$ is equivalent to the entailment $p \models_{\mathsf{S}_A} q_1, \ldots, q_n$.

Since language inclusion is decidable for NFTA[6], we leverage from an existing algorithm for this problem by Holík et al. [9] to build a complete set of inference rules and derive a proof search technique. This algorithm searches for counterexamples of the inclusion problem $\mathcal{L}(A,p) \subseteq \bigcup_{i=1}^{k} \mathcal{L}(A,q_i)$ by enumerating pairs $(r, \{s_1, \ldots, s_m\})$, where r is a state that can be reached via a sequence of transitions from

[5] Several definitions of graph automata exist but none has a decidable inclusion problem.

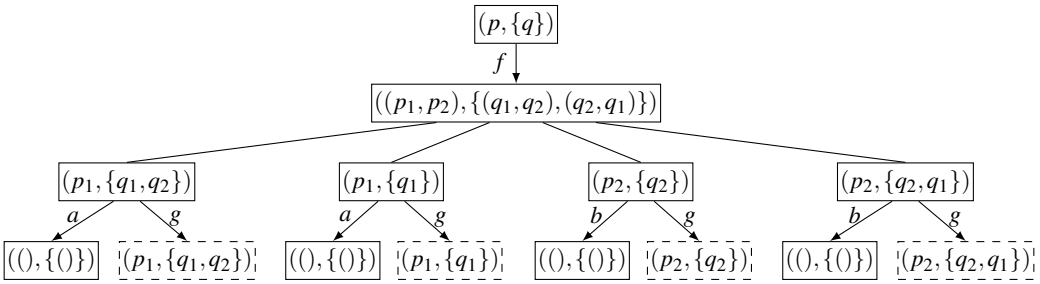
[6] See, e.g. [6, Corollary 1.7.9].

p, and $\{s_1,\ldots,s_m\}$ are all the states that can be reached via a sequence of transitions with the same function symbols, from $q_1,\ldots,q_k$, respectively. A counterexample is found when the algorithm encounters a pair $(r,\{s_1,\ldots,s_m\})$ such that there exists a rule $r \xrightarrow{a} ()$, but no rule $s_i \xrightarrow{a} ()$ for any $i \in [1,m]$.

Example 2. *Consider the tree automaton A, consisting of the transition rules:*

$$\begin{array}{ccc} p \xrightarrow{f} (p_1,p_2) & & q \xrightarrow{f} (q_1,q_2) \quad q \xrightarrow{f} (q_2,q_1) \\ p_1 \xrightarrow{g} (p_1) \quad p_1 \xrightarrow{a} () & & q_1 \xrightarrow{g} (q_1) \quad q_1 \xrightarrow{a} () \\ p_2 \xrightarrow{g} (p_2) \quad p_2 \xrightarrow{b} () & & q_2 \xrightarrow{g} (q_2) \quad q_2 \xrightarrow{b} () \end{array}$$

It is not hard to see that $\mathcal{L}(A,p) = \{f(g^n(a),g^m(b)) \mid n \geq 0, m \geq 0\}$ *and* $\mathcal{L}(A,q) = \{f(g^n(a),g^m(b)) \mid n \geq 0, m \geq 0\} \cup \{f(g^n(b),g^m(a)) \mid n \geq 0, m \geq 0\}$. *To check* $\mathcal{L}(A,p) \subseteq \mathcal{L}(A,q)$, *we start with* $(p,\{q\})$. *An execution of the top-down inclusion algorithm is pictured below:*



The algorithm performs two types of moves: transitions and split actions. The arrows labeled by symbols f,g,a *and* b *are transitions, for instance the arrow labeled by* f *takes* p *into the tuple* (p_1,p_2) *by the transition* $p \xrightarrow{f} (p_1,p_2)$ *and* $\{q\}$ *into the set of tuples* $\{(q_1,q_2),(q_2,q_1)\}$, *by the transitions* $q \xrightarrow{f} (q_1,q_2)$ *and* $q \xrightarrow{f} (q_2,q_1)$. *However, the pair* $((p_1,p_2),\{(q_1,q_2),(q_2,q_1)\})$ *is problematic because it asserts that* $\mathcal{L}(A,p_1) \times \mathcal{L}(A,p_2) \subseteq \mathcal{L}(A,q_1) \times \mathcal{L}(A,q_2) \cup \mathcal{L}(A,q_2) \times \mathcal{L}(A,q_1)$. *Using several properties of the Cartesian product [9, Theorem 1] there are multiple ways to split this proof obligation into several simpler conjunctive subgoals. If at least one split move leads to a proof, then the inital proof obligation holds. The split move used above simultaneously considers* $(p_1,\{q_1,q_2\})$, $(p_1,\{q_1\})$, $(p_2,\{q_2\})$ *and* $(p_2,\{q_2,q_1\})$, *together asserting that* $\mathcal{L}(A,p_1) \subseteq \mathcal{L}(A,q_1)$ *and* $\mathcal{L}(A,p_2) \subseteq \mathcal{L}(A,q_2)$. *The other options are: (1)* $(p_1,\{q_1,q_2\})$ $(p_1,\{q_1\})$, $(p_1,\{q_2\})$, *and* $(p_2,\{q_2,q_1\})$; *(2)* $(p_1,\{q_1,q_2\})$, $(p_2,\{q_1\})$, $(p_1,\{q_2\})$, *and* $(p_2,\{q_2,q_1\})$; *(3)* $(p_1,\{q_1,q_2\})$, $(p_2,\{q_1\})$, $(p_2,\{q_2\})$, *and* $(p_2,\{q_2,q_1\})$. *The algorithm does not expand nodes* (p,S) *with* $p \in S$, *for which inclusion holds trivially, or having a predecessor* (p,S') *with* $S' \subseteq S$ *(enclosed in dashed boxes), since any counterexample that can be found from* (p,S) *could have been discovered from* (p,S'). ■

3.2 Inference Rules and Proof Search

We now describe the set of inference rules using a Gentzen-style sequent calculus. We denote *sequents* as $\Gamma \vdash \Delta$, where Γ and Δ are sets of formulae, called *antecedent* [Γ] and *consequent* [Δ].

$$(\mathsf{R})\ \frac{\Gamma_1 \vdash \Delta_1 \ \ldots\ \Gamma_n \vdash \Delta_n}{\begin{array}{c}\Gamma \vdash \Delta \\ \vdots\ \mathbf{c} \\ \Gamma_p \vdash \Delta_p\end{array}}\ \begin{array}{l}\text{side} \\ \text{conditions}\end{array}$$

The commas in the sequents are read as set union and we avoid the enclosing brackets when writing down sequents. A sequent of the form $p(\mathbf{x}_0) \vdash q_1(\mathbf{x}_1), \ldots, q_n(\mathbf{x}_n)$, where $p, q_1, \ldots, q_n$ are predicate symbols is said to be *basic*.

An inference rule r has $\#\mathsf{r} \geq 0$ *premises* $\Gamma_i \vdash \Delta_i$, for $i = 1, \ldots, \#\mathsf{r}$ and a *conclusion* $\Gamma \vdash \Delta$. A rule with no premises is called an *axiom* and we write $\top$ for its empty premiss list. Additionally, certain inference rules may have a *pivot* $\Gamma_p \vdash \Delta_p$, which is always a sequent preceding the conclusion on the path from the root to itself. The path between the pivot and the conclusion of a rule is subject to a *pivot constraint* $\mathbf{C}$, formally defined below.

For conciseness of presentation, we define inference rule schemata R, that are possibly infinite sets of inference rules sharing the same pattern. Without entering formal details, we assume that checking whether a given inference rule instance belongs to a given schema is straightforward and that there are finitely many instances of a certain schema for a given pivot and consequent.

Definition 1. *A* derivation *is a possibly infinite tree $D = (V, v_0, S, R, P)$, where V is a set of vertices, $v_0 \in V$ is the root node, and for each vertex $v \in V$, $S(v)$ is a sequent, $R(v)$ is an inference rule schema and, if $v \neq v_0$, then $P(v)$ is the parent of v. In particular, for each $v \in V \setminus \{v_0\}$, $S(v)$ is a premiss of an instance of $R(P(v))$, whose conclusion is $S(P(v))$. A* proof *is a finite derivation, such that each leaf is the conclusion of an axiom.*

This definition strays from the classical definition of cyclic proofs, viewed as graphs, rather than trees [2, 3]. Below we recover the graph structure of a proof, by explicitly introducing backlinks:

Definition 2. *Given a derivation $D = (V, v_0, S, R, P)$, a* backlink *is a pair (u, v) with $u, v \in V$ such that u is the conclusion of the instance of $R(u)$ and v is its pivot. A* trace *is a sequence of vertices $\tau = v_1, v_2, \ldots$ such that, for all $i \geq 2$, either $v_{i-1} = P(v_i)$ or (v_{i-1}, v_i) is a backlink. A* path *of τ is any finite subsequence $\pi = v_i, \ldots, v_j$ such that $i < j$ and $v_{k-1} = P(v_k)$, for all $k \in [i+1, j]$. If, moreover, (v_j, v_i) is a backlink, then π is a* direct path.

Backlinks generalize the relation between buds and companions in [2, 3]. For a backlink (u, v), we call u a *bud* and v its *companion*, when u has no direct successors in the derivation tree. For now, the only rules introducing backlinks are axioms, hence all pivots will be companions. However we prefer to distinguish pivots from companions, for further extensions of the system with rules that may have a pivot without being axioms. Observe, moreover, that we require companions to always be ancestors of buds in the derivation. As shown in [3], this is not a restriction, because any cyclic proof can be transformed into an equivalent cyclic proof with this property[7].

To define pivot constraints formally, for a path $\pi = v_1, \ldots, v_k$ with $k \geq 2$, we denote by $R(\pi)$ the sequence $R(v_1), \ldots, R(v_{k-1})$ of inference rule schemata that are applied on π. The pivot constraint $\mathbf{C}$ of a rule schema R is a set of finite sequences of rule schemata, such that, if π is the direct path from the pivot of (the instance of) r to its conclusion, then $R(\pi) \in \mathbf{C}$.

Given a system of inductive definitions S and a set of predicates $p, q_1, \ldots, q_k \in \mathsf{Pred}$ of equal arities, a set of inference rule schemata $\mathcal{R}$ is (i) *sound* if, for any proof $D = (V, v_0, S, R, P)$ using the inference rules of $\mathcal{R}$, with $S(v_0) = p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$, we have $p \models_{\mathsf{S}} q_1, \ldots, q_n$, and (ii) *complete* if $p \models_{\mathsf{S}} q_1, \ldots, q_n$ implies the existence of a proof $D = (V, v_0, s, r, p)$ using the inference rules of $\mathcal{R}$, where $S(v_0) = p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$. The following general result will be used next, in the proof of soundness for our particular set of inference rules.

Proposition 1. *Any infinite trace of a proof D contains infinitely many direct paths.*

[7] Such proofs are said to be in *cycle normal form* [3, Definition 6.2.1].

A *strategy* is a set $\mathbf{S}$ of sequences of inference rule schemata. A sequence σ is a *prefix* of $\mathbf{S}$ if there exists another sequence τ such that $\sigma\tau \in \mathbf{S}$. A derivation D [proof] is an $\mathbf{S}$-derivation [$\mathbf{S}$-proof] if $R(\pi)$ is a prefix of $\mathbf{S}$, for each path π of D.

Algorithm 1 Proof search semi-algorithm.

data structure: $\mathsf{Node}(Seq, CList, Parent, Rule)$, where:
- *Seq* is the sequent that labels the node,
- *CList* is the list of children nodes,
- *Parent* is the link to the parent of the node,
- *Rule* is the inference rule with consequent *Seq*.

input:
- a system of inductive definitions S,
- a sequent $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$,
- a set $\mathcal{R}$ of inference rule schemata and a strategy $\mathbf{S}$

output: a proof $\mathcal{D} = (V, v_0, S, R, P)$ such that $S(v_0) = p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$

1: $\mathsf{Root} \leftarrow \mathsf{Node}(p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x}), [], nil, nil)$
2: $\mathsf{WorkList} \leftarrow \{\mathsf{Root}\}$
3: **while** $\mathsf{WorkList} \neq \emptyset$ **do**
4: remove N from WorkList and match it with $\mathsf{Node}(\Gamma \vdash \Delta, CList, P, R)$
5: let π be the path between Root and N
6: let $\mathcal{R}_N \subseteq \mathcal{R}$ be the inference rule schemata applicable on $N.Seq$ and π
7: let $\mathcal{R}_N^0 \subseteq \mathcal{R}_N$ be the subset of $\mathcal{R}_N$ with empty premiss lists
8: **if** $R(\pi) \cdot \mathsf{R}$ is a valid prefix of $\mathbf{S}$ for some $\mathsf{R} \in \mathcal{R}_N^0$ **then**
9: $N.Rule \leftarrow \mathsf{R}$
10: mark N as closed
11: **if** N not closed and $R(\pi) \cdot \mathsf{R}$ is a valid prefix of $\mathbf{S}$ for some $\mathsf{R} \in \mathcal{R}_N \setminus \mathcal{R}_N^0$ **then**
12: let r be an instance of R such that $N.Seq$ is the consequent of r
13: **for** each premiss $\Gamma' \vdash \Delta'$ of r **do**
14: $N' \leftarrow \mathsf{Node}(\Gamma' \vdash \Delta', [], N, nil)$
15: append N' to $N.CList$
16: append N' to WorkList
17: $N.Rule \leftarrow \mathsf{R}$
18: **if** $N.CList$ is empty **then** mark N as closed

Given an input sequent $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$, a set $\mathcal{R}$ of inference rules and a strategy $\mathbf{S}$, the generic proof search semi-algorithm (1) uses a worklist iteration to build a derivation of $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$. When a sequent is removed from the worklist, it chooses (non-deterministically) an inference rule and an instance whose consequent matches the current sequent, if one exists. To enhance the chance of termination, the nodes matching an axiom are considered eagerly (line 8). Since we assumed that the only rules introducing backlinks are axioms, the treatment of axioms takes also care of the coverage of buds by companions. It is not difficult to show that, if an $\mathbf{S}$-proof of the input sequent $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$ exists, then semi-algorithm (1) will output that proof.

4 Proof Rules for Entailment

The inference rules for proving entailments between inductively defined predicates are essentially of one of the three types: unfolding (U), reduction (R) and split (S). In addition, we consider two axiom

schemata, one for sequents that are valid no matter how the predicates are interpreted (AX) and another one for infinite descent (ID). We call URS and URSID the sets of inference rule schemata introduced below, without and with the (ID) rule, respectively.

A sequent $\Gamma \vdash \Delta$ stands for the (proof obligation of the) entailment $\bigwedge \Gamma \models \exists_{y \in \mathrm{FV}(\Delta) \setminus \mathrm{FV}(\Gamma)} y \,.\, \bigvee \Delta$. For a compact presentation of the rules, we omit the existential quantifiers. Moreover, we write $(\Gamma_i \vdash \Delta_i)_{i=1}^n$ for $\Gamma_1 \vdash \Delta_1 \ \ldots \ \Gamma_n \vdash \Delta_n$. Throughout the section we assume a system S of inductive definitions.

The inference rules (LU) and (RU) unfold a predicate atom $p(\mathbf{x})$ occurring in the antecedent or the consequent of a sequent $\Gamma \vdash \Delta$, respectively. By unfolding, we essentially mean the replacement of $p(\mathbf{x})$ with the bodies of the rules of its inductive definition $p(\mathbf{x}) :=_{\mathsf{S}} R_1(\mathbf{x}, \mathbf{y}_1) \mid \ldots \mid R_n(\mathbf{x}, \mathbf{y}_n)$:

$$\text{(LU)}\ \frac{(R_i(\mathbf{x}, \mathbf{y}_i),\ \Gamma \setminus p(\mathbf{x}) \vdash \Delta)_{i=1}^n}{\Gamma \vdash \Delta}\ \begin{array}{l} p(\mathbf{x}) \in \Gamma \\ \mathbf{y}_1, \ldots, \mathbf{y}_n \text{ fresh variables} \end{array}$$

$$\text{(RU)}\ \frac{\Gamma \vdash \bigwedge R_1(\mathbf{x}, \mathbf{y}_1), \ldots, \bigwedge R_n(\mathbf{x}, \mathbf{y}_n),\ \Delta \setminus p(\mathbf{x})}{\Gamma \vdash \Delta}\ \begin{array}{l} p(\mathbf{x}) \in \Delta \\ \mathbf{y}_1, \ldots, \mathbf{y}_n \text{ fresh variables} \end{array}$$

Observe that the left unfolding yields a set of premises, one for each R_i, whereas the only premiss of the right unfolding is obtained by replacing $p(\mathbf{x})$ in the consequent of the conclusion with a set of conjunctions $\bigwedge R_i$, in which the subgoal variables of each R_i are fresh.

The inference rules (RD) eliminate the constraints from both the antecedent and consequent of the conclusion. The existentially quantified variables from the consequent are eliminated using a finite subset of the set $\mathrm{Sk}(\phi, \psi_i) \stackrel{\mathsf{def}}{=} \{\theta : \mathbf{y}_i \to \mathcal{T}_\Sigma(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n) \mid \phi \models \psi\theta\}$ of substitutions, mapping the existentially quantified variables of $\psi_i(\mathbf{x}, \mathbf{y}_i)$ to Skolem terms over the free variables of $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$. These terms witness the entailments between the constraints of the antecedent and those of the consequent.

$$\text{(RD)}\ \frac{p_1(\mathbf{x}_1), \ldots, p_n(\mathbf{x}_n) \vdash \{Q_j\theta \mid \theta \in S_j\}_{j=1}^i}{\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n), p_1(\mathbf{x}_1), \ldots, p_n(\mathbf{x}_n) \vdash \{\psi_j(\mathbf{x}, \mathbf{y}_j) \wedge Q_j(\mathbf{y}_j)\}_{j=1}^k}\ \begin{array}{l} (\bigcup_{j=1}^k \mathbf{y}_j) \cap (\mathbf{x} \cup \bigcup_{i=1} \mathbf{x}_i) = \emptyset \\ \phi \models \bigwedge_{j=1}^i \exists_{y \in \mathbf{y}_j} y \,.\, \psi_j \\ \phi \not\models \bigvee_{j=i+1}^k \exists_{y \in \mathbf{y}_j} y \,.\, \psi_j \\ S_j \subseteq \mathrm{Sk}(\phi, \psi_j), \forall j \in [1, i] \end{array}$$

$$(\wedge\text{R})\ \frac{(\Gamma \vdash p_i(\mathbf{x}) \wedge Q, \Delta)_{i=1}^n}{\Gamma \vdash p_1(\mathbf{x}) \wedge \ldots \wedge p_n(\mathbf{x}) \wedge Q, \Delta}$$

Because (RD) can create conjunctions of predicate atoms with the same arguments, every application of (RD) will be followed by a cleanup of the premiss consequents, using a right introduction rule ($\wedge$R).

To understand the relation between proofs and tree automata language inclusion checking, we view a pair $(p, \{q_1, \ldots, q_n\})$ in the top-down antichain algorithm (Example 2) as a sequent $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$. In the algorithm of Example 2, the (LU), (RU) and (RD) rules are combined in a single transition move. This is natural because the transition rules of tree automata are controlled uniquely by the function symbols labeling the root of the current input term. Since function symbols can only be compared via equality, the constraints within the predicate rules of a tree automata match unambiguously. For instance, we have $x \approx f(x_1, x_2) \models \exists y_1 \exists y_2 \,.\, x \approx g(y_1, y_2)$ if and only if f and g are the same function symbol, in which case the only substitution witnessing the validity of the entailment is $\theta(x_i) = y_i$, for $i = 1, 2$. However, when considering generic first-order constraints, matching amounts to discovering non-trivial substitutions that prove an entailment between existentially quantified formulae in the logic in which the constraints are written. Moreover, the matching step implemented by the (RD) rule is crucial for proving completeness of the set of inference rules, by generalizing from the simple case of tree automata constraints and discovering general properties of the set of constraints that allow matching to be complete (§5.1).

Next, the splitting rules (SP) generalize the split moves of the inclusion algorithm (Example 2) by

breaking a sequent $p_1(\mathbf{x}_1), \ldots, p_n(\mathbf{x}_n) \vdash Q_1(\mathbf{x}_1, \ldots, \mathbf{x}_n), \ldots, Q_n(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ into basic sequents.

$$\text{(SP)}\ \frac{\left(p_{\bar{\imath}_j}(\mathbf{x}) \vdash \{q^{\ell}_{\bar{\imath}_j}(\mathbf{x}) \mid \ell \in [k],\ f_j(\overline{Q}_\ell) = \bar{\imath}_j\}\right)_{j=1}^{n^k}}{p_1(\mathbf{x}_1), \ldots, p_n(\mathbf{x}_n) \vdash Q_1(\mathbf{x}_1, \ldots, \mathbf{x}_n), \ldots, Q_k(\mathbf{x}_1, \ldots, \mathbf{x}_n)} \begin{array}{l} \forall i,j \in [1,n] \cdot \mathbf{x}_i \cap \mathbf{x}_j = \emptyset,\ \bar{\imath} \in [1,n]^{n^k} \\ Q_i = \bigwedge_{j=1}^{n} q^i_j(\mathbf{x}_j), \overline{Q}_i = \langle q^i_1, \ldots, q^i_n \rangle \\ \mathcal{F}(\overline{Q}_1, \ldots, \overline{Q}_k) = \{f_1, \ldots, f_{n^k}\} \end{array}$$

Given tuples $\{\overline{Q}_1, \ldots, \overline{Q}_k\}$ consisting of $n \geq 1$ predicate atoms each, a *choice function* f maps each $\overline{Q}_i$ into an index $f(\overline{Q}_i) \in [1,n]$ corresponding to a position in the tuple. We denote by $\mathcal{F}(\overline{Q}_1, \ldots, \overline{Q}_k)$ the set of choice functions, of cardinality $n^k \leq n^{\|\mathsf{Pred}\|^n}$. Given a tuple $\bar{\imath} \in [1,n]^{n^k}$, associating a value in $[1,n]$ to each choice function $f \in \mathcal{F}(\overline{Q}_1, \ldots, \overline{Q}_k)$, there exists an application of (SP) generating n^k premises with antecedent $p_{\bar{\imath}_j}(\mathbf{x}), j \in [1,n^k]$ and consequent consisting of all predicate atoms $q^{\ell}_{\bar{\imath}_j}(\mathbf{x}), \ell \in [1,k]$ obtained from predicates at position $\bar{\imath}_j$ in the tuples $\overline{Q}_\ell$ which are mapped to $\bar{\imath}_j$ by f_j.

Finally, we introduce two axiom schemata. First, the rules (AX) close the current branch of the proof if the sequent can be proved using a decision procedure for the underlying constraint logic, which treats the predicates as uninterpreted boolean functions. For the tree automata language inclusion algorithm, this rule is similar to encountering a pair (p,S), with $p \in S$.

$$\text{(AX)}\ \frac{\top}{\Gamma \vdash \Delta}\ \wedge\Gamma \models \exists_{y \in \mathrm{FV}(\Delta) \setminus \mathrm{FV}(\Gamma)}\ y \cdot \vee\Delta \qquad \text{(ID)}\ \frac{\begin{array}{c}\top \\ \vdots\ \mathsf{URSID}^* \cdot \mathrm{LU} \cdot \mathsf{URSID}^* \end{array}}{\Gamma\theta \vdash \Delta'\theta}\ \begin{array}{l}\theta \text{ flat injective substitution} \\ \Delta \subseteq \Delta'\end{array}$$

Second, the infinite descent rules (ID) are the only rules to introduce backlinks, from the conclusion $\Gamma\theta \vdash \Delta'\theta$ to a predecessor (pivot) vertex labeled with $\Gamma \vdash \Delta$. The pivot condition $\mathsf{URSID}^* \cdot \mathrm{LU} \cdot \mathsf{URSID}^*$ requires that a vertex labeled with a rule (LU) occurs on the direct path between the pivot and the conclusion. As we explain next, this condition suffices to guarantee soundness of URSID.

We recall that the inclusion algorithm of Example 2 stops expanding a branch in the search tree whenever it has discovered a pair (p,S') that has a predecessor (p,S), with $S \subseteq S'$. In analogy, the (ID) rules close the current branch of the derivation, when the last sequent $\Gamma\theta \vdash \Delta'\theta$ has a predecessor $\Gamma \vdash \Delta$ with $\Delta \subseteq \Delta'$, modulo a substitution θ. For technical reasons, related to the soundness of URSID, we consider only flat injective such substitutions, when applying (ID).

Example 3. *Consider the system S_A, corresponding to the tree automaton from Example 2:*

$$\begin{array}{ll} p(x) & :=_{\mathsf{S}_A} x \approx f(x_1,x_2), p_1(x_1), p_2(x_2) \\ p_1(x) & :=_{\mathsf{S}_A} x \approx g(x_1), p_1(x_1) \mid x \approx a \\ p_2(x) & :=_{\mathsf{S}_A} x \approx g(x_1), p_2(x_1) \mid x \approx b \end{array} \qquad \begin{array}{ll} q(x) & :=_{\mathsf{S}_A} x \approx f(x_1,x_2), q_1(x_1), q_2(x_2) \\ & \quad \mid\ x \approx f(x_1,x_2), q_2(x_1), q_1(x_2) \\ q_1(x) & :=_{\mathsf{S}_A} x \approx g(x_1), q_1(x_1) \mid x \approx a \\ q_2(x) & :=_{\mathsf{S}_A} x \approx g(x_1), q_2(x_1) \mid x \approx b \end{array}$$

Part of the proof for $p(x) \vdash q(x)$ is shown below. Note the similarities with the algorithm of Example 2, whose transition moves correspond to subtrees of the proof rooted in an (LU) *node whose children are successively labeled with* (RU) *or* (RD).

$$\mathrm{SP}\frac{[\dagger]\ p_1(x) \vdash q_1(x), q_2(x) \quad p_1(x) \vdash q_1(x) \quad p_2(x) \vdash q_2(x) \quad p_2(x) \vdash q_2(x), q_1(x)}{\mathrm{RD}\dfrac{p_1(x_1), p_2(x_2) \vdash q_1(x_1) \wedge q_2(x_2), q_2(x_1) \wedge q_1(x_2)}{\mathrm{RU}\dfrac{\begin{array}{r} x \approx f(x_1,x_2), p_1(x_1), p_2(x_2) \vdash x \approx f(y_1,y_2) \wedge q_1(y_1) \wedge q_2(y_2), \\ x \approx f(y_1,y_2) \wedge q_2(y_1) \wedge q_1(y_2) \end{array}}{\mathrm{LU}\dfrac{x \approx f(x_1,x_2), p_1(x_1), p_2(x_2) \vdash q(x)}{p(x) \vdash q(x)}}}$$

$$\dfrac{\dfrac{\dfrac{}{x \approx a \vdash x \approx a, q_2(x),\ x \approx g(y_1) \wedge q_1(y_1)}\,\text{AX}}{x \approx a \vdash q_1(x), q_2(x)}\,\text{RU} \qquad \dfrac{\dfrac{\dfrac{}{p_1(x_1) \vdash q_1(x_1), q_2(x_1)\ (*)}\,\text{ID}}{x \approx g(x_1), p_1(x_1) \vdash x \approx a, x \approx b,\ x \approx g(y_1) \wedge q_1(y_1),\ x \approx g(y_1) \wedge q_2(y_1)}\,\text{RD}}{x \approx g(x_1), p_1(x_1) \vdash x \approx a, q_2(x),\ x \approx g(y_1) \wedge q_1(y_1)}\,\text{RU}}{x \approx g(x_1), p_1(x_1) \vdash q_1(x), q_2(x)}\,\text{RU}$$

$$\dfrac{x \approx a \vdash q_1(x), q_2(x) \qquad x \approx g(x_1), p_1(x_1) \vdash q_1(x), q_2(x)}{[\dagger]\ p_1(x) \vdash q_1(x), q_2(x)\ (*)}\,\text{LU}$$

The split move of the algorithm corresponds to the single node labeled (SP)*, whose consequent consists of conjunctions of* $\overline{Q}_1 = (q_1, q_2)$ *and* $\overline{Q}_2 = (q_2, q_1)$*. The set of choice functions is* $\mathcal{F}(\overline{Q}_1, \overline{Q}_2) = \{f_1 = \{(\overline{Q}_1, 1), (\overline{Q}_2, 1)\}, f_2 = \{(\overline{Q}_1, 1), (\overline{Q}_2, 2)\}, f_3 = \{(\overline{Q}_1, 2), (\overline{Q}_2, 1)\}, f_4 = \{(\overline{Q}_1, 2), (\overline{Q}_2, 2)\}\}$*. Out of the 16 index choice tuples for* $\mathcal{F}(\overline{Q}_1, \overline{Q}_2)$*, only* $(1,1,1,2)$*,* $(1,1,2,2)$*,* $(1,2,1,2)$ *and* $(1,2,2,2)$ *are relevant. To obtain the above proof, we chose* $\bar{\imath} = (1,1,2,2)$*. We mark with* $(*)$ *the pivot required for the application of the* (ID) *rule above. Observe that the pivot constraint is satisfied because the pivot is the conclusion of a* (LU) *rule.* ■

4.1 Soundness of URSID

We establish soundness of the URSID set of inference rules, by first proving the local soundness of URS. More precisely, for every rule that is an instance of a rule schema $\mathsf{R} \in \mathsf{URS}$, the validity of all premises implies the validity of the conclusion. For technical reasons that will become clear in a moment, Lemma 1 below proves the equivalent contrapositive statement.

Let S be a fixed system, U a universe and I an interpretation, throughout this section. For a formula ϕ and an assignment $\mathcal{X}$, that maps each predicate symbol p occurring in ϕ to a set $\mathcal{X}(p) \subseteq \mathsf{U}^{\#(p)}$, we denote by $[\![\phi]\!]_{\mathcal{X}}$ the set of valuations ν of $\mathrm{FV}(\phi)$, such that $(\mathsf{U}, I, \nu) \models \phi$, when each predicate atom $p(\mathbf{x})$ is true if and only if $\nu(\mathbf{x}) \in \mathcal{X}(p)$. For conciseness, we omit the formal definition.

Moreover, given a set F consisting of predicate atoms $q_1(\mathbf{x}_1), \ldots, q_n(\mathbf{x}_n)$ and several other formulae, not containing predicate atoms, for a valuation ν of $\mathrm{FV}(F)$, we denote by $\mathcal{M}(F, \nu)$ the multiset of stage numbers $sn(q_1, \nu(\mathbf{x}_1)), \ldots, sn(q_n, \nu(\mathbf{x}_n)) \in \mathbb{N}$. It is easy to see that the antecedent of any sequent $\Gamma \vdash \Delta$ that occurs in a URS derivation consists only of predicate atoms and constraints, thus $\mathcal{M}(\Gamma, \nu)$ is defined, for each valuation ν of $\mathrm{FV}(\Gamma)$. Before giving the local soundness result (Lemma 1), we briefly recall the definition of the Manna-Dershowitz multiset order for natural numbers:

Definition 3 ([8])**.** *Given multisets* $M, N \subseteq \mathbb{N}$*, we write* $N \leq^{\dagger} M$ *if and only if either* $M = N$*, or there exists a non-empty finite multiset* $X \subseteq M$ *and a (possibly empty) multiset* Y*, where for all* $y \in Y$ *there exists* $x \in X$ *such that* $y < x$ *and* $N = (M \setminus X) \cup Y$*.*

It is known that the $\leq^{\dagger}$ order is well-founded, since $\leq$ is well-founded on $\mathbb{N}$ [8].

Lemma 1. *For each instance of an inference rule schema* $\mathsf{R} \in \mathsf{URS}$*, with conclusion* $\Gamma \vdash \Delta$ *and premises* $(\Gamma_i \vdash \Delta_i)_{i=1}^{n}$ *and for each valuation* ν *of* $\mathrm{FV}(\Gamma)$ *such that* $\nu \in [\![\bigwedge \Gamma]\!]_{\mu\mathsf{S}} \setminus [\![\exists_{y \in \mathrm{FV}(\Delta) \setminus \mathrm{FV}(\Gamma)}\ y\ .\ \bigvee \Delta]\!]_{\mu\mathsf{S}}$*, there exists a valuation* $\nu_i \in [\![\bigwedge \Gamma_i]\!]_{\mu\mathsf{S}} \setminus [\![\exists_{y \in \mathrm{FV}(\Delta_i) \setminus \mathrm{FV}(\Gamma_i)}\ y\ .\ \bigvee \Delta_i]\!]_{\mu\mathsf{S}}$ *such that* $\mathcal{M}(\Gamma, \nu) \geq^{\dagger} \mathcal{M}(\Gamma_i, \nu_i)$*, for some* $i \in [1, n]$*. Moreover, if* $\mathsf{R} = \mathrm{LU}$ *then* $\mathcal{M}(\Gamma, \nu) >^{\dagger} \mathcal{M}(\Gamma_i, \nu_i)$*.*

We introduce a relation between valuations and write $\nu \triangleright \nu'$ if and only if ν is a countermodel of the conclusion of an inference rule and ν' is the countermodel of one of its premises, whose existence is asserted by Lemma 1.

To tackle the soundness of URSID, we must show that a countermodel cannot be indefinitely propagated along an infinite trace through the proof, going via backlinks. A first technical point that must be addressed is the extension of the $\triangleright$ relation to the backlinks introduced by the (ID) inference rules. Let $\Gamma \vdash \Delta$ be the conclusion and $\Gamma' \vdash \Delta'$ be the pivot of an (ID) rule. Now suppose that $\nu \in [\![\bigwedge \Gamma]\!]_{\mu\mathsf{S}} \setminus [\![\exists_{y \in \mathrm{FV}(\Delta) \setminus \mathrm{FV}(\Gamma)} \, y \, . \, \bigvee \Delta]\!]_{\mu\mathsf{S}}$ is a countermodel valuation of the conclusion. By the side condition, we have $\Gamma = \Gamma'\theta$ and $\Delta = \Delta''\theta$, for a set $\Delta'' \subseteq \Delta'$, where θ is a flat and injective substitution. Since the restriction of θ to the set $\mathrm{FV}(\Gamma'\theta) \cup \mathrm{FV}(\Delta'\theta)$ is bijective, it has an inverse and $\nu \circ \theta^{-1}$ is a countermodel for $\Gamma' \vdash \Delta'$, i.e. $\nu \circ \theta^{-1} \in [\![\bigwedge \Gamma']\!]_{\mu\mathsf{S}} \setminus [\![\exists_{y \in \mathrm{FV}(\Delta') \setminus \mathrm{FV}(\Gamma')} \, y \, . \, \bigvee \Delta']\!]_{\mu\mathsf{S}}$[8]. Moreover, because θ is flat, we have $\mathcal{M}(\Gamma, \nu) = \mathcal{M}(\Gamma', \nu \circ \theta)$.

Now suppose, for a contradiction, that there exists a proof $D = (V, v_0, S, R, P)$ for an invalid sequent $S(v_0) = \Gamma_0 \vdash \Delta_0$. Then we can build an infinite trace $\tau = u_0, u_1, \ldots$ through D, and a sequence of valuations $\nu_0 \triangleright \nu_1 \triangleright \ldots$, such that ν_i is a countermodel for $S(v_i) = \Gamma_i \vdash \Delta_i$, for all $i \geq 0$. Moreover, by Lemma 1 and the extended definition of $\triangleright$ for backlinks, we have that $\mathcal{M}(\Gamma_i, \nu_i) \geq^{\dagger} \mathcal{M}(\Gamma_{i+1}, \nu_{i+1})$, for all $i \geq 0$. But, by Proposition 1, τ must contain an infinite number of direct paths. Since, by the pivot condition of (ID), any direct path has at least one application of (LU), by the second point of Lemma 1, we obtain an infinitely decreasing sequence of multisets $\mathcal{M}(\Gamma_{j_1}, \nu_{j_1}) >^{\dagger} \mathcal{M}(\Gamma_{j_2}, \nu_{j_2}) >^{\dagger} \ldots$, where $j_1 < j_2 < \ldots$, thus contradicting the well-foundedness of $\geq^{\dagger}$. Hence there is no countermodel to start with and the entailment $\bigwedge \Gamma_0 \models \exists_{y \in \mathrm{FV}(\Delta_0) \setminus \mathrm{FV}(\Gamma_0)} \, y \, . \, \bigvee \Delta_0$ must hold.

Theorem 2. *For any* URSID *proof* $D = (V, v_0, S, R, P)$*, such that* $S(v_0) = \Gamma \vdash \Delta$*, the entailment* $\bigwedge \Gamma \models \exists_{y \in \mathrm{FV}(\Delta) \setminus \mathrm{FV}(\Gamma)} \, y \, . \, \bigvee \Delta$ *holds.*

5 Relative Completeness of URSID

As previously discussed, the undecidability of the entailment problem (Theorem 1) excludes the possibility of having a complete set of inference rules in which every valid entailment has a finite proof. In this section, we identify a class of inductive definition systems for which URSID is complete and the entailment problem is decidable. Unlike with most decidable fragments of first (second) order logic, we do not impose syntactic restrictions on the form of the inductive definitions in the system. Rather, we give three conditions that must be satisfied by all quantifier-free constraints in the inductive rules. Moreover, we tackle the decidability of these conditions in the Herbrand interpretation of first-order logic and give several complexity upper bounds for the problem of checking whether a given system complies with them. Before giving the completeness proof under these restrictions, we suggest a stronger but easy-to-check syntactic criterion.

5.1 Restricting the Set of Constraints

The following definitions introduce sufficient conditions that ensure completeness of URSID. Effectively checking these conditions for a given inductive system is subject to the existence of a decision procedure for the $\exists^*\forall^*$ fragment of the logic in which the constraints are written, i.e. the set of prenex normal form sentences of the form $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m \, . \, \phi$, where ϕ is a quantifier-free boolean combination of equality and interpreted predicate atoms. For first-order logic with the Herbrand interpretation, this problem, known as *disunification*, has been shown decidable in [7], with tighter complexity bounds given in [12]. In the rest of this section, we assume a fixed universe U and interpretation I.

The first restriction requires that, given any models for the subgoals of an inductive rule, it must be possible to find an all-encompassing model that also satisfies the constraint of the rule. This restriction

[8] Because $(\mathsf{U}, I, \nu) \models \phi \iff (\mathsf{U}, I, \nu \circ \theta^{-1}) \models \phi\theta$, for an arbitrary formula ϕ and a bijective substitution θ.

is necessary because lifting it leads, in general, to the undecidability of the entailment problem, as it is the case for tree automata with equality and disequality constraints[9].

Definition 4. *A system of inductive definitions* S *is* non-filtering *if and only if, for every rule* $(\{\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n),q_1(\mathbf{x}_1),\ldots,q_n(\mathbf{x}_n)\},p(\mathbf{x}))\in\mathsf{S}$*, for all* $i\in[1,n]$ *and all* $\mathbf{v}_i\in\mu\mathsf{S}(q_i)$*, there exists a valuation* ν*, such that* $\nu(\mathbf{x}_i)=\mathbf{v}_i$ *and* $(\mathsf{U},I,\nu)\models\phi$*.*

Unfortunately, this condition is undecidable for Herbrand models, as showed by the following:

Lemma 2. *The problem "given an inductive system* S*, is* S *non-filtering?" is undecidable in the Herbrand interpretation.*

This negative result suggests adopting a stronger condition. Namely, we require that, for each rule $(\{\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n),q_1(\mathbf{x}_1),\ldots,q_n(\mathbf{x}_n)\},p(\mathbf{x}))$, the formula $\forall\mathbf{x}_1\ldots\forall\mathbf{x}_n\exists\mathbf{x}.\,\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n)$ is valid, or equivalently, that $\exists\mathbf{x}_1\ldots\exists\mathbf{x}_n\forall\mathbf{x}.\,\neg\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n)$ is unsatisfiable. Because each constraint ϕ is a conjunction of equalities $s\approx t$ and disequalities $\neg(s\approx t)$ between terms, $\neg\phi$ is a disjunction of equalities and disequalities, trivially in conjunctive normal form. Since the satisfiability of the formulae $\exists\mathbf{x}\forall\mathbf{y}.\,\phi(\mathbf{x},\mathbf{y})$, with ϕ in conjunctive normal form, is NP-complete[10], the above validity problem is in co-NP.

Example 4. *The* S_A *system from Example 3 is non-filtering. If, for instance, the rules for p were changed to* $p(x)\coloneqq_{\mathsf{S}_A} x\approx f(x_1,x_2)\wedge x_1\approx x_2,p_1(x_1),p_2(x_2)$*, then* S_A *would become filtering, as all the subgoals models for which the values of* x_1 *and* x_2 *differ would be rejected by the new predicate rule.* ■

The second restriction guarantees that all constraints can be eliminated from a sequent, by instantiating the free variables of the consequent, that do not occur in the antecedent, using finitely many substitutions that map onto the subgoal variables from the antecedent. Given a sequent such as, e.g. $\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n),p_1(\mathbf{x}_1),\ldots,p_n(\mathbf{x}_n)\vdash\exists\mathbf{y}_1\ldots\exists\mathbf{y}_m.\,\psi(\mathbf{x},\mathbf{y}_1,\ldots,\mathbf{y}_m)\wedge q_1(\mathbf{y}_1)\wedge\ldots\wedge q_m(\mathbf{y}_m)$ if the entailment $\phi\models\exists\mathbf{y}_1\ldots\exists\mathbf{y}_m.\,\psi$ is valid, the (RD) inference rule replaces it with a finite number of sequents $p_1(\mathbf{x}_1),\ldots,p_n(\mathbf{x}_n)\vdash\{q_1(\mathbf{y}_1\theta)\wedge\ldots\wedge q_m(\mathbf{y}_m\theta)\mid\theta\in S\}$, where $\phi\models\psi\theta$, for each substitution θ.

This elimination of constraints from sequents is generally sound, but incomplete. The above entailment is valid if $\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n)\models\psi'(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_m)$, where ψ' is obtained from ψ by replacing each $y\in\mathbf{y}_1\cup\ldots\cup\mathbf{y}_m$ with a Skolem function symbol $f_y(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n)$ not occurring in ϕ or ψ[11]. A complete inference rule would have to consider all possible interpretations of these Skolem witnesses. This is impossible in general, as their definitions are not bound to any particular form.

To ensure completeness of URSID, we require that these functions are defined as flat substitutions ranging over $\mathbf{x}\cup\bigcup_{i=1}^n\mathbf{x}_i$. This condition ensures that there are finitely many possible interpretations of these Skolem witnesses.

Definition 5. *A system of inductive definitions* S *has the* finite instantiation *property if and only if, for any two constraints* $\phi(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n)$ *and* $\psi(\mathbf{x},\mathbf{y}_1,\ldots,\mathbf{y}_m)$ *from* S*, with goal variables* $\mathbf{x}$ *and subgoal variables* $\bigcup_{i=1}^n\mathbf{x}_i$ *and* $\bigcup_{j=1}^m\mathbf{y}_j$*, respectively, the set* $\mathrm{Sk}(\phi,\psi)\stackrel{\text{def}}{=}\{\theta:\bigcup_{i=1}^m\mathbf{y}_i\to\mathcal{T}_\Sigma(\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n)\mid\phi\models\psi\theta\}$ *is finite. Moreover,* S *has the* finite variable instantiation (fvi) *property iff for all* $j\in[1,m]$ *there exists* $i\in[1,n]$ *such that* $\mathbf{y}_j\theta=\mathbf{x}_i$*, for each* $\theta\in\mathrm{Sk}(\phi,\psi)$*.*

Example 5. *The* S_A *inductive system from Example 3 has the fvi property. Take, for instance, the constraints* $\phi\equiv x\approx f(x_1,x_2)$ *and* $\psi\equiv x\approx f(y_1,y_2)$*. The entailment* $\phi\models\exists y_1\exists y_2.\,\psi$ *is witnessed by a single substitution* θ *with* $\theta(x_1)=y_1$ *and* $\theta(x_2)=y_2$*, which means that* $\mathrm{Sk}(\phi,\psi)=\{\theta\}$*.* ■

[9] See [6, Theorem 4.2.10].

[10] See [12, Theorem 5.2].

[11] We assume w.l.o.g. that these function symbols belong to the signature.

We give an upper bound for the complexity of the problem whether a given inductive system has the fvi property, in the Herbrand interpretation of constraints. It is unclear, for now, whether this bound can be tightened, because the exact complexity of the satisfiability of equational problems is unknown[12].

Lemma 3. *The problem "given an inductive system* S*, does* S *have the fvi property?" is in* NEXPTIME *in the Herbrand interpretation. If there exists a constant* $K > 0$*, independent of the input, such that for each constraint* $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$*, with goal variables* $\mathbf{x}$ *and subgoal variables* $\bigcup_{i=1}^{n} \mathbf{x}_i$*, respectively, we have* $\|\mathbf{x}_i\| \leq K$*, then the problem is in* NP.

The third and last condition required for the completeness of URSID is also related to the elimination of constraints from sequents. Intuitively, we do not allow two constraints to overlap, having at least one model in common, without one entailing the other.

Definition 6. *An inductive system* S *is* non-overlapping *if and only if, for any two constraints* $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$ *and* $\psi(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m)$ *in* S*, with goal variables* $\mathbf{x}$ *and subgoal variables* $\bigcup_{i=1}^{n} \mathbf{x}_i$ *and* $\bigcup_{j=1}^{m} \mathbf{y}_j$ *respectively,* $\phi \wedge \psi$ *is satisfiable only if* $\phi \models \exists \mathbf{y}_1 \ldots \exists \mathbf{y}_m \,.\, \psi$.

For a non-overlapping system, if $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n) \wedge \psi(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m)$ is a satisfiable conjunction of constraints, then the formulae $\exists \mathbf{x}_1 \ldots \exists \mathbf{x}_n \,.\, \phi$ and $\exists \mathbf{y}_1 \ldots \mathbf{y}_m \,.\, \psi$ are equivalent. It unclear, for now, whether this condition can be lifted by considering, e.g. disjoint conjunctive minterms consisting of constraints and their negations. This is mainly because introducing negative constraints might affect the non-filtering property of the system (Definition 4).

Lemma 4. *The problem "given a first order inductive system* S*, is* S *non-overlapping?" is in* NP *in the Herbrand interpretation.*

Example 6. *The* S_A *inductive system from Example 3 is non-overlapping. Consider, for instance, the constraints* $x \approx f(x_1, x_2)$ *and* $x \approx f(y_1, y_2)$*. Then* $x \approx f(x_1, x_2) \wedge x \approx f(y_1, y_2)$ *is satisfiable and* $x \approx f(x_1, x_2) \models \exists y_1 \exists y_2 \,.\, x \approx f(y_1, y_2)$ *holds. However, if we take the constraints* $x \approx f(x_1, x_2)$ *and* $x \approx g(y_1)$*, then* $x \approx f(x_1, x_2) \wedge x \approx g(y_1)$ *is unsatisfiable.* ■

5.1.1 A Syntactic Completeness Criterion

The three conditions above (Definitions 4, 5, 6) are of technical nature and meant for machine rather than human checking. That is, given a particular instance of the entailment problem, we show the existence of algorithms that answer yes/no to the question whether URSID is complete for that particular instance — in case of a negative answer, the proof search semi-algorithm 1 is not bound to terminate. However, it is desirable to also have a stronger but easy-to-check criterion. The following definition introduces such a criterion, and the next lemma proves that every inductive defnition system that complies with it has a decidable entailment problem.

Given terms t and u, a substitution θ is a *unifier* for t and u if and only if $t = u\theta$. Observe that the relation "t and u have a unifier" is a preorder but not an equivalence. We assume, in the rest of this section, the Herbrand interpretation of function symbols.

Definition 7. *An inductive definition system* S *is said to be* rooted *if and only if each constraint is of the form* $\bigwedge_{x \in \mathbf{x}} x \approx t_x$*, where* $\mathbf{x}$ *are the goal variables, each* t_x *is a term using subgoal variables only and, moreover, for any two terms* t_x *and* u_x *that occur in different constraints* $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$ *and* $\psi(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m)$*, with subgoal variables* $\bigcup_{i=1}^{n} \mathbf{x}_i$ *and* $\bigcup_{j=1}^{m} \mathbf{y}_m$*, respectively, if* t_x *and* u_x *have a unifier* θ*, then for each* $j \in [1, m]$ *there exists* $i \in [1, n]$ *such that* $(\mathrm{FV}(u) \cap \mathbf{y}_j)\theta = \mathrm{FV}(t) \cap \mathbf{x}_i$.

[12] Converting a formula into CNF requires exponential time at most, thus NEXPTIME is an upper bound.

Observe that rooted systems are strictly more expressive than the inductive definitions of tree automata. For instance, the rooted system from Example 3 defines non-regular sets of terms.

Example 7. *Consider the system S_r below:*

$$\begin{array}{rcl} n(x_1,x_2) & :=_{\mathsf{S}_r} & x_1 \approx a \wedge x_2 \approx b \mid x_1 \approx f(y_1) \wedge x_2 \approx f(y_2), n(y_2,y_1) \\ m(x) & :=_{\mathsf{S}_r} & x \approx g(y_1,y_2), n(y_1,y_2) \\ e(x_1,x_2) & :=_{\mathsf{S}_r} & x_1 \approx a \wedge x_2 \approx b \mid x_1 \approx f(y_1) \wedge x_2 \approx f(y_2), o(y_1,y_2) \\ o(x_1,x_2) & :=_{\mathsf{S}_r} & x_1 \approx f(y_1) \wedge x_2 \approx f(y_2), e(y_1,y_2) \\ p(x) & :=_{\mathsf{S}_r} & x \approx g(y_1,y_2), e(y_1,y_2) \mid x \approx g(y_1,y_2), o(y_2,y_1) \end{array}$$

It is easy to see that S_r is rooted and, moreover, that $\mu\mathsf{S}_r(m) = \mu\mathsf{S}_r(p) = \{g(f^{2n}(a), f^{2n}(b)) \mid n \geq 0\} \cup \{g(f^{2n+1}(b), f^{2n+1}(a)) \mid n \geq 0\}$, which is not a regular tree language. Below we sketch a URSID *proof for the sequent $m(x) \vdash p(x)$:*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z_1 \approx a, z_2 \approx b \vdash z_1 \approx a \wedge z_2 \approx a, z_1 \approx f(u_1) \wedge z_2 \approx f(u_2) \wedge o(u_1,u_2), o(z_2,z_1)}\,\mathrm{AX}\,(\top)}{z_1 \approx a, z_2 \approx b \vdash e(z_1,z_2), o(z_2,z_1)}\,\mathrm{RU} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{n(v_1',v_2') \vdash e(v_1',v_2'), o(v_1',v_2')\ (*)}\,\mathrm{ID}\,(\top)}{\vdots}}{[\dagger]\ n(v_1,v_2) \vdash e(v_2,v_1), o(v_1,v_2)}}{z_1 \approx f(v_1), z_2 \approx f(v_2), n(v_1,v_2) \vdash z_1 \approx a \wedge z_2 \approx b, z_1 \approx f(w_1) \wedge z_2 \approx f(w_2) \wedge o(w_1,w_2), z_2 \approx f(w_1') \wedge z_1 \approx f(w_2') \wedge e(w_1',w_2')}\,\mathrm{RU}}{z_1 \approx f(v_1), z_2 \approx f(v_2), n(v_1,v_2) \vdash e(z_1,z_2), o(z_2,z_1)}\,\mathrm{LU}$$

$$\dfrac{\dfrac{\dfrac{[\surd]\ n(z_1,z_2) \vdash e(z_1,z_2), o(z_2,z_1)\ (*)}{x \approx g(z_1,z_2), n(z_1,z_2) \vdash x \approx g(y_1,y_2) \wedge e(y_1,y_2), x \approx g(y_1,y_2) \wedge o(y_2,y_1)}\,\mathrm{RD}}{x \approx g(z_1,z_2), n(z_1,z_2) \vdash p(x)}\,\mathrm{RU}}{m(x) \vdash p(x)}\,\mathrm{LU}$$

The derivation rooted in the $(\dagger)$ is a symmetric copy of the derivation rooted in the $(\surd)$ node and ending with $(\dagger)$. As usual, we mark the conclusion of (ID) *and its companion with $(*)$. The pivot constraint is satisfied because the pivot is the conclusion of an* (LU) *rule.* ■

The following shows that rooted systems always non-filtering, non-overlapping and have the fvi property. As shown in the next section, this provides a sufficient completeness criterion and, moreover, an argument for the decidability of the entailment problem within this class of systems.

Proposition 2. *Any rooted system is non-filtering, non-overlapping and has the fvi property.*

5.2 Completeness of URSID

We prove that URSID is complete for entailments between predicates in inductive systems that are non-filtering, non-overlapping and have the fvi property (§5.1). A derivation is said to be *maximal* if it cannot be extended by an application of an inference rule, and *irreducible* if it cannot be rewritten into a smaller derivation of the same sequent by an (ID) application. Note that the proof search semi-algorithm (1) generates only irreducible derivations, because (ID) is always applied before any other inference rules.

A derivation D is *structured* if, on each path of D, between any two consecutive applications of (LU) there exists an application of (RD). Intuitively, unstructured derivations constitute poor candidates for proofs. For instance, a derivation consisting only of applications of (LU) rules will only grow the size of

the left-hand sides of the sequents, without making progress towards $\top$ or a countermodel. Observe that each subtree of a structured derivation is also structured. We denote by $\mathfrak{D}(\Gamma \vdash \Delta)$ the set of irreducible, maximal and structured derivations $D = (V, v_0, S, R, P)$ whose root nodes are labeled by $S(v_0) = \Gamma \vdash \Delta$.

Lemma 5. *If* S *has the fvi property, then the following hold:*
1. *any irreducible and structured derivation is finite, and*
2. *for any sequent* $\Gamma \vdash \Delta$*, the set* $\mathfrak{D}(\Gamma \vdash \Delta)$ *is finite.*

Next, we prove an invariant on the shape of the sequents occurring in a proof of a basic sequent.

Definition 8. *A set* $F = \{\phi_1, \ldots, \phi_n, q_1(\mathbf{x}_1), \ldots, q_m(\mathbf{x}_m)\}$ *is* tree-shaped *if and only if* $\phi_1, \ldots, \phi_n$ *are constraints,* $q_1(\mathbf{x}_1), \ldots q_m(\mathbf{x}_m)$ *are predicate atoms, and there exist trees* $t_1, \ldots, t_k$ *such that:*
- *each node labeled with a constraint* $\phi_i(\mathbf{y}, \mathbf{y}_1, \ldots, \mathbf{y}_n)$ *in some tree* t_ℓ*,* $\ell \in [1,k]$ *has exactly n children and for all* $j \in [1,n]$*, the* j*-th child is labeled either (i) with a constraint whose goal variables are* $\mathbf{y}_j$*, or (ii) with a predicate atom* $q_k(\mathbf{y}_j)$*, and*
- *a predicate atom* $q_i(\mathbf{x}_i)$ *may only be a leaf in a tree* t_j*, for some* $j \in [1,k]$*.*

Tree-shaped sets can be uniquely represented by trees labeled with formulae, thus we use sets of trees instead of sets of formulae interchangeably.

Lemma 6. *Given an inductive system* S *and the predicate atoms* $p(\mathbf{x}), q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$*, in every sequent* $\Gamma \vdash \Delta$ *such that* $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x}) \rightsquigarrow \Gamma \vdash \Delta$*,* Γ *is a tree-shaped set and* Δ *consists of finite conjunctions of tree-shaped sets.*

The following lemma characterizes the cases in which the root of a derivation is labeled by an invalid sequent, from which a countermodel can be extracted. This is crucial in establishing our completeness result (Theorem 3). We write $\Gamma \vdash \Delta \rightsquigarrow \Gamma' \vdash \Delta'$ iff $\Gamma' \vdash \Delta'$ occurs in a derivation from $\mathfrak{D}(\Gamma \vdash \Delta)$.

Lemma 7. *Given a non-filtering and non-overlapping inductive definition system* S *with the fvi property, the predicate atoms* $p(\mathbf{x}), q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$*, and a sequent* $\Gamma \vdash \Delta$ *such that* $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x}) \rightsquigarrow \Gamma \vdash \Delta$*, if every derivation* $\mathcal{D} \in \mathfrak{D}(\Gamma \vdash \Delta)$ *contains a leaf labeled with sequent with empty consequent, then there exists a valuation* $\nu \in [\![\bigwedge \Gamma]\!]_{\mu\mathsf{S}} \setminus [\![\exists_{y \in \mathrm{FV}(\Delta) \setminus \mathrm{FV}(\Gamma)} y \cdot \bigvee \Delta]\!]_{\mu\mathsf{S}}$.

The following theorem proves that URSID is complete and provides a proof search strategy.

Theorem 3. *Given a non-filtering, non-overlapping inductive system* S *with the fvi property, let* $p, q_1, \ldots, q_n$ *be predicates symbols occurring in* S*. Then the entailment* $p \models_{\mathsf{S}} q_1, \ldots, q_n$ *holds only if the sequent* $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$ *has an* $\mathbf{S}$*-proof with the set of inference rules* URSID*, where* $\mathbf{S}$ *is defined by the regular expression* $(\mathrm{LU} \cdot \mathrm{RU}^* \cdot \mathrm{RD} \cdot \wedge \mathrm{R}^* \cdot \mathrm{SP}?)^* \cdot \mathrm{LU}? \cdot \mathrm{RU}^* \cdot (\mathrm{AX} \mid \mathrm{ID})$.

Proof. Since S is non-filtering and non-overlapping, and moreover, it has the fvi property and also $p \models_{\mathsf{S}} q_1, \ldots, q_n$, by Lemma 7, there exists a finite maximal, structured and irreducible derivation $D \in \mathfrak{D}(p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x}))$ which does not contain a leaf labeled by a sequent with empty consequent. But then, no node in D is labeled by a sequent with empty consequent, because all descendants of such a node must have empty consequents as well.

We first show that this derivation is actually a proof (i.e. all its leaves are labeled with $\top$). Suppose there exists a leaf that is not labeled with $\top$. This means that the leaf is labeled by a sequent $\Gamma \vdash \Delta$, where $\Delta \neq \emptyset$. Let π be the path in D leading to this leaf. Since D is a maximal derivation, π cannot be extended any further by the application of an inference rule. Assume that the last inference rule applied on π belongs to the schema R and its parent is labeled by $\Gamma' \vdash \Delta'$. Since $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x}) \rightsquigarrow \Gamma' \vdash \Delta'$, by Lemma 6, Γ' is a tree-shaped set and Δ' is a set of conjunctions of tree-shaped sets. Also, R cannot be (AX) or (ID), because then the leaf would be labeled by $\top$. We do a case split based on R:

1. (LU) Then $\Gamma \vdash \Delta$ is of the form $R, \Gamma' \setminus r(\mathbf{y}) \vdash \Delta'$, where $(R, r(\mathbf{y})) \in \mathsf{S}$ is an inductive rule. If Δ' contains at least one predicate atom, we can still apply (RU), which contradicts the fact that D is maximal. Otherwise, because $\Delta' \neq \emptyset$ consists of conjunctions of tree-shaped sets and does not contain any predicate atoms, it must be the case that Δ' contains only non-trivial conjunctions, obtained from previous applications of (RU) or (RD). We distinguish the following cases:
   - $\Gamma' \setminus r(\mathbf{y}) = \emptyset$. Then we can apply (RD) to the sequent $R \vdash \Delta'$ and extend D, a contradiction.
   - $\Gamma' \setminus r(\mathbf{y}) \neq \emptyset$. If (R) is the first occurrence of a rule (LU) then it must be the case that $\Gamma' \setminus r(\mathbf{y}) = \emptyset$, which contradicts our assumption. Then there must have been a previous application of (LU) on π, and, because D is structured, (RD) must have been applied between them. Therefore, since Γ' is tree-shaped, $\Gamma' \setminus r(\mathbf{x})$ can only contain predicates, because the constraints introduced by (LU) are always eliminated by (RD). Then we can apply (LU) and extend D, a contradiction.
2. (RU) Then $\Gamma \vdash \Delta$ is of the form $\Gamma' \vdash \{\bigwedge R_i(\mathbf{y}, \mathbf{z}_i)\}_{i=1}^{m}, \Delta' \setminus r(\mathbf{y})$, where $r(\mathbf{y})$ is a predicate atom and $r(\mathbf{y}) :=_{\mathsf{S}} R_1 \mid \ldots \mid R_m$. If $\Delta' \setminus r(\mathbf{y})$ contains at least one predicate, we can apply (RU) and extend D, contradiction. Otherwise, because $\Delta' \neq \emptyset$ consists of conjunctions over tree-shaped sets and it does not contain any predicates, then it must be the case that Δ' contains only non-trivial conjunctions, obtained from previous applications of (RU) or (RD). We distinguish the cases:
   - Γ' contains a predicate atom. Then we can apply (LU) and extend D, contradiction.
   - Γ' does not contain predicate atoms. Because Γ' is tree-shaped and D is structured, Γ' can only contain a constraint with no subgoal variables. Then we can apply (RD) and extend D, contradiction.
3. (RD) Then $\Gamma \vdash \Delta$ is of the form $r_1(\mathbf{y}_1), \ldots, r_m(\mathbf{y}_m) \vdash Q_1(\mathbf{y}_1, \ldots, \mathbf{y}_m), \ldots, Q_k(\mathbf{y}_1, \ldots, \mathbf{y}_m)$ and we can apply (LU) – or even ($\wedge$R), (RU) or (SP) if possible – to $\Gamma \vdash \Delta$, which means that we can still extend π, leading to a contradiction.
4. ($\wedge$R) $\Gamma \vdash \Delta$ is of the form $\Gamma \vdash r(\mathbf{y}) \wedge Q, \Delta''$. Since we only apply ($\wedge$R) as cleanup after (RD), Γ only contains predicate atoms and Q is a conjunction of predicate atoms. Then we can continue to apply ($\wedge$R) if $r(\mathbf{y}) \wedge Q$ or any member of Δ'' contains some conjunction $s_1(\mathbf{z}) \wedge \ldots \wedge s_k(\mathbf{z})$, or apply (LU), (RU), or (SP), leading to a contradiction.
5. (SP) Then $\Gamma \vdash \Delta$ is of the form $r(\mathbf{y}) \vdash s_1(\mathbf{y}), \ldots, s_m(\mathbf{y})$ and we can apply (LU) or (RU) to $\Gamma \vdash \Delta$, which means that we can still extend π and leads to a contradiction.

We will now show that the sequence of inference rules fired on each maximal path in D is captured by the strategy $\mathbf{S}$. Let π be an arbitrary maximal path in D. Since D is a maximal derivation, π cannot be extended any further by the application of an inference rule. W.l.o.g. we assume that the first application of (LU) is not immediately preceded by an application of (RU) — otherwise, one can obtain the same sequent by first applying (LU) before any (RU). The proof goes by induction on the number $N \geq 1$ of basic sequents that occur on π. If $N = 1$, then the only basic sequent $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$ occurs on the first position of π. In this case (SP) is never applied on π, because its premiss is also basic sequent, contradicting with $N = 1$. We distinguish two cases:

1. If (LU) is not applied on π, the only possibility is to apply $\mathrm{AX} \in \mathbf{S}$ to $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$, thus ending the path. Otherwise, (LU) is enabled, which contradicts the maximality of π.
2. Else, since (LU) is applied on π, then it must be applied in the beginning, because only (LU) and (RU) are applicable on $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$ and we assumed that no instance of (RU) immediately precedes (LU). Assume that the first rule applied on π is:

$$\text{(LU)}\ \frac{\Gamma' \vdash \Delta'}{p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})}$$

   Then $(\Gamma', p(\mathbf{x})) \in \mathsf{S}$ is an inductive rule and (LU) cannot be applied again without applying (RD) first, due to the assumption that D is structured. Since $\Delta' = \{q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})\}$, we can now apply

(RU). Because (SP) is never applied on π, either (AX) or (RD) can be applied next. In the first case, we obtain $R(\pi) \in \mathrm{LU} \cdot \mathrm{RU} \cdot \mathrm{AX} \in \mathbf{S}$. In the second case, if $n = 1$ in the premiss of (RD), since S has the fvi property, we obtain that the premiss of (RD) is a basic sequent, contradicting our assumption. Then it must be the case that $n > 1$, and now (RD) is not applicable any longer, because the number of predicate atoms will always be bigger than the number of subgoal variables in the constraint from the antecedent. The only possibilities for continuation are then (AX), (ID), ($\wedge$R), (LU) and (RU). However, ($\wedge$R) is applicable only a finite number of times, equal to the number of conjunctions of predicate atoms with the same arguments from the consequent, (RU) can also only be applied a finite number of times, equal to the number of singleton predicate atoms from the antecedent, and (LU) is applicable at most once, because (RD) is no longer applicable. In both cases, π is not maximal, because (LU) is enabled. Then the only possibility is to end the path by (AX) or (ID), obtaining $\mathrm{LU} \cdot \mathrm{RU} \cdot \mathrm{RD} \cdot \wedge\mathrm{R}^* \cdot \mathrm{LU}? \cdot \mathrm{RU}^* \cdot (\mathrm{AX} \mid \mathrm{ID}) \subseteq \mathbf{S}$.

If $N > 1$, let $\pi = \tau \cdot \rho$, where ρ starts with the second occurrence of a basic sequent in π. As before, the first occurrence is the initial sequent $p(\mathbf{x}) \vdash q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})$. Then the inference rule applied on the last vertex of τ is either (SP) or (RD). In the latter case, the sequent's antecedent consists of a single predicate atom. In the former case, there is an application of (RD) optionally followed by several applications of ($\wedge$R) preceding the last vertex of τ, and consider now the conclusion of the last application of (RD) on a vertex in τ. As argued before, the antecedent of this conclusion is a predicate rule of S with goal $r(\mathbf{y})$, introduced by a previous application of (LU). Since, between the premiss of this (LU) instance and the conclusion of (RD), the antecedent of the sequents is unchanged, the only possibility is that (RU) has been used between them, thus $R(\pi) \in \mathrm{LU} \cdot \mathrm{RU}^* \cdot \mathrm{RD} \cdot \wedge\mathrm{R}^* \cdot \mathrm{SP}? \cdot R(\rho)$. By the inductive hypothesis, we have $R(\rho) \in \mathbf{S}$, thus $R(\pi) \in (\mathrm{LU} \cdot \mathrm{RU}^* \cdot \mathrm{RD} \cdot \wedge\mathrm{R}^* \cdot \mathrm{SP}?) \cdot \mathbf{S} \subseteq \mathbf{S}$, as required. □

The proof search semi-algorithm (1) only explores irreducible derivations. If executed with the strategy $\mathbf{S}$ from Theorem 3, these derivations are also structured. By Lemma 5 (1), irreducible and structured derivations are finite, thus every execution of the semi-algorithm is guaranteed to terminate. If, moreover, the input inductive system S is ranked, non-filtering, non-overlapping and has the fvi property, then URSID is complete and semi-algorithm (1) is a decision procedure for the entailment problems of S.

6 Conclusions and Future Work

We present a cyclic proof system for entailments between inductively defined predicates written using first-order logic, based on Fermat's principle of Infinite Descent. The soundness of this principle is coined by a semantic restriction on the constraints of the inductive system, that asks that models generated by unfoldings decrease in a well-founded domain. On the other hand, completeness is applicable under three additional semantic restrictions on the set of constraints. In general, all these restrictions are decidable, with computational complexities that depend on the logical fragment in which the constraints of the inductive system are written.

References

[1] Berardi, S., Tatsuta, M.: Classical system of Martin-Löf's inductive definitions is not equivalent to cyclic proof system. In: Foundations of Software Science and Computation Structures (FoSSaCS). pp. 301–317 (2017)

[2] Brotherston, J.: Cyclic proofs for first-order logic with inductive definitions. In: Proc. of the 14th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. pp. 78–92 (2005)

[3] Brotherston, J.: Sequent Calculus Proof Systems for Inductive Definitions. Ph.D. thesis, University of Edinburgh (November 2006)

[4] Brotherston, J., Simpson, A.: Sequent calculi for induction and infinite descent. J. Log. Comput. 21(6), 1177–1216 (2011)

[5] Bussey, W.H.: Fermat's method of infinite descent. The American Mathematical Monthly 25(8), 333–337 (1918)

[6] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: Tree automata techniques and applications (2005), http://www.grappa.univ-lille3.fr/tata

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

[8] Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Commun. ACM 22(8), 465–476 (Aug 1979)

[9] Holík, L., Lengál, O., Simácek, J., Vojnar, T.: Efficient inclusion checking on explicit and semi-symbolic tree automata. In: ATVA 2011, Proc. pp. 243–258 (2011)

[10] Iosif, R., Serban, C.: An entailment checker for separation logic with inductive definitions. In: 18th International Workshop on Automated Verification of Critical Systems (2018)

[11] Madhusudan, P., Qiu, X., Stefanescu, A.: Recursive proofs for inductive tree data-structures. SIGPLAN Not. 47(1), 123–136 (Jan 2012)

[12] Pichler, R.: On the complexity of equational problems in cnf. Journal of Symbolic Computation 36, 235 – 269 (2003)

[13] Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: Proc. of LICS. pp. 55–74 (2002)

[14] Rowe, R.N.S., Brotherston, J.: Realizability in cyclic proof: Extracting ordering information for infinite descent. In: Automated Reasoning with Analytic Tableaux and Related Methods. pp. 295–310. Springer International Publishing, Cham (2017)

[15] Serban, C.: Automated Reasoning in Separation Logic with Inductive Definitions. Ph.D. thesis, Université Grenoble Alpes (May 2018), https://hal.archives-ouvertes.fr/tel-01856199

[16] Sipser, M.: Introduction to the Theory of Computation. PWS Publishing Company, Boston, MA, USA (1997)

[17] Stump, A., Oe, D., Reynolds, A., Hadarean, L., Tinelli, C.: SMT proof checking using a logical framework. Formal Methods in System Design 42(1), 91–118 (2013)

[18] Walukiewicz, I.: Completeness of Kozen's axiomatisation of the propositional mu-calculus. Information and Computation 157(1), 142 – 182 (2000)