A Unified View of Induction Reasoning for First-Order Logic

Sorin Stratulat

A Unified View of Induction Reasoning for First-Order Logic

Sorin Stratulat

LITA, Department of Computer Science
Université de Lorraine
Ile du Saulcy, Metz, 57000, FRANCE
sorin.stratulat@univ-lorraine.fr

Abstract

Induction is a powerful proof technique adapted to reason on sets with an unbounded number of elements. In a first-order setting, two different methods are distinguished: the conventional induction, based on explicit induction schemas, and the implicit induction, based on reductive procedures. We propose a new cycle-based induction method that keeps their best features, i.e., performs local and non-reductive reasoning, and naturally fits for mutual and lazy induction. The heart of the method is a proof strategy that identifies in the proof script the subset of formulas contributing to validate the application of induction hypotheses. The conventional and implicit induction are particular cases of our method.

1 Introduction

Induction is a most successful proof technique to reason on unbounded data structures like naturals and lists. Already known by the ancient greeks and rediscovered by Pascal and Fermat at the second half of the seventeenth century, it became a precious formal tool for mathematicians to such an extent that Poincaré considered it at the beginning of the twentieth century as the mathematical reasoning *par excellence* [41]. This is also the case for computer science. The fundamental notion of computable function can be defined in many equivalent models of computation, among which the Turing machines [54] and the (μ-)recursive functions [31]. There is a strong link between recursion and induction; for example, the correctness of primitive recursive functions is proved by structural induction on naturals.

After the inception of computers, McCarthy saw an interest for its mechanization and coded the recursion induction method [37] into Lisp. Later on, Burstall [15] showed the importance of structural schemata-based induction to verify properties about recursively defined data structures. Since then, a lot of 'proof by induction' methods have been proposed and contributed to many successful stories about the validation and verification of (critical) user specifications. Particular remark should be made on the Knuth-Bendix saturation-based completion procedure [32] that opened the way to rewrite-based automated reasoning methods like the inductionless induction [23, 25, 36, 39] and other reductive induction methods as implicit [33, 44] and cyclic induction [46]. Today, it is hardly imaginable a modern theorem prover not integrating 'proof by induction' features.

The most general induction principles are the Noetherian induction and its contrapositive version, the 'Descente Infinie' (or Infinite Descent) induction. They allow to prove the validity of a property ϕ for any element from a potentially infinite poset $(\mathcal{E}, <)$, provided that $<$ is a *well-founded* ordering that excludes the occurrence of any infinite strictly decreasing sequence of elements. The Noetherian induction principle can be formally stated as follows:

Noetherian induction. $(\forall m \in \mathcal{E}, (\forall k \in \mathcal{E}, k < m \Rightarrow \phi(k)) \Rightarrow \phi(m)) \Rightarrow \forall p \in \mathcal{E}, \phi(p)$.

Thanks to the well-founded property of the ordering, the assumptions $\phi(k)$, called *induction hypotheses* (IHs), can be soundly applied in the proof of the *induction conclusion* $\phi(m)$.

'Descente Infinie' induction. $(\forall m \in \mathcal{E}, \neg\phi(m) \Rightarrow (\exists k \in \mathcal{E}, k < m \wedge \neg\phi(k))) \Rightarrow \forall p \in \mathcal{E}, \phi(p)$.

The soundness proof of a 'Descente Infinie' induction principle is by contradiction. If there exists an element $m_0 \in \mathcal{E}$ such that $\neg\phi(m_0)$, according to the induction principle, there is a smaller element m_1 such that $\neg\phi(m_1)$, for which there is an even smaller element m_2 such that $\neg\phi(m_2)$, and so on. In this way, an infinite strictly descending sequence of elements of $\mathcal{E}$ is built. This contradicts the well-foundedness property of the ordering.

In a first-order setting, two useful classes of Noetherian/'Descente Infinie' induction instances are distinguished, for which the elements of $\mathcal{E}$ are i) (vectors of) terms, and ii) (first-order) formulas. The first class includes the conventional induction methods, based on induction schemas. They explicitly define the induction hypotheses which are linked to the induction conclusions in order to be used in further derivations. The information justifying the soundness property is *locally* embedded inside the induction schemas, hence their natural integration into deductive, sequent-based inference systems in terms of *induction rules*. During a proof, it may happen to define useless IHs or to ask for crucial IHs that are not yet defined. The case of mutual induction, for which instances of a formula are used as IHs in the proof of other formulas, and viceversa, is hardly manageable with non-mutual recursion and leads to more technical and complex proofs.

Implicit induction methods are part of the second class. Formulated in a 'Descente Infinie' induction form, a set of formulas are true if for any false formula there is a smaller one. These methods are *lazy* since the IHs are defined when needed. In fact, any formula can be applied as IH to another formula as long as it is smaller, hence implicit induction fits well for mutual induction. Since any formula from a proof script is susceptible to be an IH, a typical soundness proof makes a *global* analysis of the formulas from the proof script. Moreover, their implementation is based on *reductive* procedures that transform formula instances into strictly smaller or, sometimes, smaller or equal ones in *every* proof step.

Goals. Our main objectives are two-fold: i) to understand the relations between term- and formula-based induction principles in order to bridge the gap between the underlying proof methods, and ii) to free the inductive reasoning from computation and make it more effective.

Contributions. We analyse and shed a new light on the relations between the two classes of induction principles that culminates with an instantiation result which shows that any term-based induction principle can be represented as a formula-based induction principle. In addition, we propose a new formula-based induction method that gathers the best features of the existing methods. It is shown that the conventional and implicit induction are instances of it.

Importance. From a theoretical point of view, the new induction method uniquely synthesizes the overall usage of induction reasoning in first-order logic. From a practical point of view, it naturally performs lazy and mutual induction. The heart of the method is a proof strategy that identifies subsets of formulas from the proof script, called *cycles*, able to discharge IHs. Therefore, the soundness analysis is *local*, at cycle level, such that a different induction ordering may be used for each cycle. Finally, the method is reductive-free, allowing for less restrictive specifications, more efficient implementations and proof certification processes.

Related works. Musser [39] was the first to compare conventional with inductionless induction methods. Since then, a lot of effort was put into clarifying the relations between explicit

and implicit induction principles, [16, 22, 27, 30, 40, 56] being among the most notable. Other studies have been conducted to reduce the gap between them. Protzen [42] proposed a proof strategy to perform lazy induction on particular explicit induction proofs. Kapur and Subramaniam [29] devised a method that extends schemata-based induction to deal with a special class of mutually defined functions. Courant [18] identified a class of implicit induction inference systems for which the proofs can be reconstructed into conventional induction proofs. Reddy [44] designed implicit induction inference rules that look similar to schemata-based induction rules. This feature has been implemented by subsequent formula-based induction systems [3, 6, 12] to generate more compact and readable proofs. Instantiation results similar to ours have been achieved more recently for particular cases of *cyclic* proof systems. Sprenger and Dam [46] have shown the equivalence of two Gentzen-style proof systems for first-order μ-calculus with explicit approximations; one of them integrates local term-based induction rules, while the other lacks such induction rules. In turn, the second can build finite ω-regular proof trees for which the induction reasoning is argued by an external global induction discharge condition associated to the proof structure. In the same line, Brotherston and Simpson [14] compared two classical first-order sequent calculus proof systems; the local induction is performed using conventional induction together with a rule that deals with a class of mutual inductive definitions. In this context, they showed that any proof using local induction arguments can be represented as a proof using global induction arguments and conjectured that the other direction also holds.

Structure of the paper. The paper has five sections. After the introduction, Section 2 overviews the term- and formula-based induction proof methods. The features of each presented method are analysed by the means of different proofs of a very simple running example. Section 3 introduces the new reductive-free formula-based induction method and its properties. Section 4 gives an example that cannot be directly proved neither by explicit nor by implicit induction techniques, but it can be successfully proved with the new method. It also gives some statistics about the certification process of implicit induction proofs involved in the validation process of a non-trivial application. The representation of proofs in the non-reductive form proposed by the new method can dramatically diminish the number of reductive constraints to be checked. Section 5 concludes and outlines perspectives.

2 First-order Induction Reasoning

Induction reasoning is helpful when establishing properties about programs and specifications built from recursively defined domains and data structures. The properties of interest are *inductive*, i.e., they are valid only in distinguished term-generated models. However, it is hardly imaginable to perform inductive reasoning exclusively with reasoning techniques working only for inductive models. Indeed, the current inductive theorem provers mix them with *deductive* techniques that can validate properties in all models.

2.1 Basic notions

Let $\mathcal{F}$ (resp. $\mathcal{P}$) be a ranked alphabet of function (resp. predicate) symbols and $\mathcal{V}$ a countable set of variables. $\mathcal{T}(\mathcal{F}, \mathcal{V})$ denotes the set of terms over $\mathcal{F}$ and $\mathcal{V}$, and $\mathcal{T}(\mathcal{F})$ its subset of ground terms. The expression $P(t_1, \ldots, t_n)$ is an *atom*, where $P \in \mathcal{P}$ is an n-ary predicate and $t_1, \ldots, t_n$ are terms. The set $\mathcal{A}(\mathcal{P}, \mathcal{F}, \mathcal{V})$ (resp. $\mathcal{A}(\mathcal{P}, \mathcal{F})$) denotes the set of atoms over $\mathcal{P}$, $\mathcal{F}$ and $\mathcal{V}$ (resp. $\mathcal{P}$ and $\mathcal{F}$). Let $\mathcal{L}$ represent a decidable set of first-order formulas over $\mathcal{A}$. We denote by Ax the *axioms* that build a specification, consisting of formulas from $\mathcal{L}$. New terms and formulas

can be obtained by substitution operations, representing finite mappings between variables and terms, of the form $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. A substitution is a *renaming* substitution if all substituting terms are variables. In addition, if any substituting variable is the same as the substituted variable, the substitution is an *identity* substitution. Given a substitution σ and a term t (resp. formula ϕ), $t\sigma$ (resp. $\phi\sigma$), sometimes written (t, σ) (resp. (ϕ, σ)), denotes the *instance* of t (resp. ϕ) by the substitution σ. Given two substitutions σ_1 and σ_2, by $\psi(\sigma_1\sigma_2)$ we denote $(\psi\sigma_1)\sigma_2$, for any term or formula ψ; $\sigma_1\sigma_2$ means that σ_1 is *composed* with σ_2. We are referring to *ground* substitutions if the mapping terms are ground.

Semantically, given a first-order structure $\mathcal{S}$ with domain $\mathcal{D}$, we write $f^{\mathcal{S}}$ to denote the interpretation of the symbol $f \in \mathcal{F}$ in $\mathcal{S}$. The variables are interpreted by a *valuation* (total) function from variables of $\mathcal{V}$ to $\mathcal{D}$. It can be extended to non-variable terms as follows: if $t(s_1, \ldots, s_n)$ is a term such that $t^{\mathcal{S}} : D^n \to D$ is the interpretation of t, its interpretation is obtained by replacing each of its i) function symbol f by $f^{\mathcal{S}}$, and ii) variable by its valuation. Every n-ary predicate symbol from $\mathcal{P}$ is interpreted as an n-ary relation on D. We write $\mathcal{S} \models_\zeta \phi$ for a formula ϕ that is true in $\mathcal{S}$ using the valuation function ζ. Any *interpretation* $< \mathcal{S}, \zeta >$ satisfying Ax is a *model* of Ax. A formula ϕ is a *deductive consequence* of a set of formulas Φ, denoted by $\Phi \models \phi$, if ϕ holds in all models of Ax whenever ψ holds in all models of Ax, for any $\psi \in \Phi$. A formula is *deductively valid* w.r.t. Ax iff it is a deductive consequence of Ax.

Deductive relation. Let $\mathbb{P}(\mathcal{L})$ denote recursive sets over $\mathcal{L}$. The recursively enumerable *deductive relation* $\vdash \subseteq \mathbb{P}(\mathcal{L}) \times \mathcal{L}$ satisfies the properties:

- if $\phi \in Ax$, then $Ax \vdash \phi$,
- if $Ax \vdash \phi$ and $Ax \subseteq Ax'$, then $Ax' \vdash \phi$,
- if $Ax \vdash \phi$ and $Ax \cup \{\phi\} \vdash \phi'$, then $Ax \vdash \phi'$, and
- if $Ax \vdash \phi$, then $Ax \vdash \phi\sigma$, for any substitution σ.

In practice, the first-order (deductive) systems that implement the deductive relation are normally *sound*, i.e., $Ax \vdash \phi$ implies $Ax \models \phi$, and *complete*, i.e., $Ax \models \phi$ implies $Ax \vdash \phi$, for any formula ϕ.

2.2 First-order induction principles

Inductive relations. We are interested in the case when Ax has term-based (Herbrand) models, i.e., whose valuation functions transform variables into ground terms. A formula ϕ is an *inductive consequence* of the axioms if ϕ holds in all Herbrand models of Ax. The *inductive theory* of Ax consists of all inductive consequences of Ax. In general, it is neither decidable, nor semi-decidable. For various reasons which depend not only on the nature and form of the axioms but also on the user's intuition, monotonicity behavior and operational feasibility criteria [57], only a non-empty subset of the Herbrand models of Ax is considered. For example, when Ax is a set of universally quantified Horn clauses with equality, it is convenient to reason on the unique *initial* (minimal) model of Ax. The choice of the model subset influences the way the variables are instantiated during the induction proofs.

In the paper, we consider that such subset, denoted by $\mathcal{M}$, exists and it is already fixed before performing any induction reasoning. To simplify the presentation, we assume that all variables are universally quantified. A formula ϕ is a $\mathcal{M}$ - *consequence* (or just consequence) of a set of formulas Φ, denoted by $\Phi \models_{\mathcal{M}} \phi$, if $\mathcal{S} \models_\zeta \phi$ whenever $\mathcal{S} \models_\zeta \psi$, for any $\psi \in \Phi$ and

model $< \mathcal{S}, \zeta >$ from $\mathcal{M}$. A formula ϕ is $\mathcal{M}$-*valid* (or just valid), denoted by $\models_{\mathcal{M}} \phi$ iff it is a consequence of Ax. The two notions of consequence relation may coincide for particular cases; for example, a positive clause ϕ is an inductive consequence of a set of universally quantified Horn clauses with equality iff ϕ is valid in their initial model [24].

A formula ϕ is *false*, denoted by $\not\models_{\mathcal{M}} \phi$, if it is not valid. Any false formula has (or contains) at least one false ground instance, called *counterexample*. It can be easily shown that, for any formula ϕ and set of formulas Ψ, if $\Psi \models_{\mathcal{M}} \phi$ and $\not\models_{\mathcal{M}} \phi$ then it exists a formula $\psi \in \Psi$ such that $\not\models_{\mathcal{M}} \psi$.

Sufficient completeness. The axioms define very often functions based on constructors that are *sufficiently complete*. In this case, the set $\mathcal{F}$ is split into defined function ($\mathcal{DF}$) and constructor ($\mathcal{C}$) symbols. In addition, the models from $\mathcal{M}$ are constructor models, i.e., whose valuation functions transform variables into ground constructor terms from $\mathcal{T}(\mathcal{C})$.

A function symbol $f \in \mathcal{DF}$ is *sufficiently complete* if any ground term of the form $f(\bar{t})$ is deductively equivalent to a constructor ground term from $\mathcal{T}(\mathcal{C})$, where $\bar{t}$ is a term vector of the form $(t_1, \ldots, t_n)$ with $t_i \in \mathcal{T}(\mathcal{C}), \forall i \in [1..n]$.

Induction orderings. The induction orderings are assumed to satisfy essential properties related to well-foundedness and stability under substitutions, as defined below. Well-foundedness implies the existence of minimal elements.

Lemma 1. *Let $(\mathcal{E}, <)$ be a well-founded poset. For any non-empty subset of $\mathcal{E}$, there is a minimal element.*

Proof. By contradiction, assume that $\mathcal{E}$ has a non-empty subset $\mathcal{E}'$ without minimal elements, i.e., for each element of $\mathcal{E}'$ there is a smaller element in $\mathcal{E}'$. Let x_1 be an arbitrary element of $\mathcal{E}'$. Since it is not minimal, it exists an element x_2 of $\mathcal{E}'$ smaller than x_1. The same reasoning is performed on x_2, and so on, to finally build an infinite strictly decreasing sequence of elements of $\mathcal{E}'$. Contradiction because the ordering is well-founded. □

In a first-order setting, the set $\mathcal{E}$ may contain an infinite number of elements that can be either ground terms or ground formulas. An effective reasoning can be pursued only on finite descriptions of them by the means of terms and/or formulas with variables. Hopefully, this reasoning can be projected and reused to the ground level when needed if the ordering also satisfies the property of *stability under substitutions*, i.e., a smaller comparison result between two terms (resp. two formulas) does not change after the instantiation of the two terms (resp. two formulas) with the same substitution, for any substitution.

As a running example, let us prove the conjecture $x + 0 = x$, for all natural x, using the axioms $0 + x = x$, for any natural x, and $S(x) + y = S(x + y)$, for any naturals x and y, that define the addition symbol '+' over the naturals based on the constructor symbols 0 and S. '=' is the only predicate symbol and the deductive system is based on equational logic [4]. '+' is sufficiently complete and the unique model from $\mathcal{M}$ is the initial model of the axioms since it fits well to reason over naturals.

Let n be an arbitrary natural, hence it is either 0 or a successor of another natural n'. In the first case, $0 + 0 = 0$ is deductively valid as an instance of the first axiom. The second case, $S(n') + 0 = S(n')$, is deductively equivalent to $S(n' + 0) = S(n')$ using the second axiom. The IH $n' + 0 = n'$ can help to replace $S(n' + 0)$ by $S(n')$ in order to obtain an identity and finish the proof.

The sound use of $n' + 0 = n'$ can be argued by two different instances of the Noetherian/'Descente Infinie' induction principles, whether the set $\mathcal{E}$ consists of ground terms or ground

formulas. The underlying induction orderings over (vector of) terms and formulas are generically denoted by $<_t$ and $<_f$, respectively.

Term-based induction. Let ϕ be a first-order formula defining a property to be checked for a non-empty set of term vectors $\mathcal{E}$. If $\bigcup_{\overline{k}\in\mathcal{E},\overline{k}<_t\overline{m}}\{\phi(\overline{k})\} \models_{\mathcal{M}} \phi(\overline{m})$, for any term vector $\overline{m} \in \mathcal{E}$, then $\forall \overline{p} \in \mathcal{E}, \models_{\mathcal{M}} \phi(\overline{p})$.

Soundness proof. By contradiction, if $\phi(\overline{m})$ is assumed to be false for some $\overline{m} \in \mathcal{E}$, we define the non-empty subset $\mathcal{E}' \subseteq \mathcal{E}$ representing all term vectors for which ϕ is false. By Lemma 1, $\mathcal{E}'$ has minimal term vectors, and let $\overline{m'}$ be such a term vector. The term-based induction rule can be applied to prove the existence of a term vector from $\mathcal{E}'$ smaller than $\overline{m'}$, so contradiction. □

A well-known term-based induction principle is the Peano induction principle: to prove a formula $P(x)$ for any natural x, it is enough to prove it for $P(0)$ and $P(S(x'))$, where x' is a fresh natural variable. The IH $P(x')$ can be soundly used in the proof of $P(S(x'))$ because $x' < S(x')$, where $<$ is the 'smaller' relation over the naturals. In our example, $P(x)$ stands for $x + 0 = x$, so $P(n')$ is the IH for proving $P(S(n'))$.

On the other hand, the IH $n' + 0 = n'$ can also be legitimated by formula-based induction principles.

Formula-based induction. Let $\mathcal{E}$ be a non-empty set of first-order formulas. If for any formula $\delta \in \mathcal{E}, \bigcup_{\gamma\in\mathcal{E},\gamma<_f\delta}\{\gamma\} \models_{\mathcal{M}} \delta$ then $\forall \rho \in \mathcal{E}, \models_{\mathcal{M}} \rho$.

Soundness proof. The Noetherian induction principle presented in the introductory part is instantiated such that $\mathcal{E}$ consists of first-order formulas and the predicate ϕ is the identity relation, i.e., $\phi(x) = x$, for any formula $x \in \mathcal{E}$. □

Formulated into a 'Descente Infinie' setting, this principle states that a potentially infinite set of first-order formulas are true if for any false formula there is another formula which is also false but smaller w.r.t. the well-founded ordering. The proof of this statement is by contradiction: we assume a false formula in $\mathcal{E}$. We consider the non-empty subset set $\mathcal{E}'$ of $\mathcal{E}$ consisting of all the false formulas from $\mathcal{E}$. By Lemma 1, there exists a minimal false formula in $\mathcal{E}'$ for which there is no smaller false formula, so contradiction.

The formula-based induction principle can be applied if the proof of $x + 0 = x$ has been generated using some reductive system such that, for any natural x', $x' + 0 = x'$ is smaller than $S(x' + 0) = S(x')$ which is in turn smaller than $S(x') + 0 = S(x')$.[1] The set $\mathcal{E}$ consists of all formulas encountered in the proof script of $x + 0 = x$, i.e., $\{x + 0 = x, 0 + 0 = 0, S(x') + 0 = S(x'), S(x' + 0) = S(x'), S(x') = S(x')\}$. The soundness proof uses a *reductio ad absurdum* reasoning at the ground level. By contradiction, we assume that $\mathcal{E}$ has a counterexample. Since the ordering is well-founded, there is a minimal counterexample of it. It can only be an instance of $x + 0 = x$ because the deductively valid formulas and the formulas deductively equivalent with but greater than $x + 0 = x$ cannot have minimal counterexamples. Let it be $n' + 0 = n'$, for some natural n'. n' should be of the form $S(n'')$ since $0 + 0 = 0$ is deductively true. In the proof script, $S(x) + 0 = S(x)$ is transformed into the smaller equality $S(x + 0) = S(x)$, for any natural x, so $S(n'' + 0) = S(n'')$ is a smaller counterexample thanks to the 'stability under substitutions' property of the ordering. In the next step of the proof script, $S(x + 0) = S(x)$ is

[1] Such an ordering over equalities can be the multiset extension of the rpo ordering based on the increasing precedence over the function symbols 0, S, and $+$.

transformed into an identity using $x + 0 = x$, for any x. Instantiating x with n'', we conclude that $n'' + 0 = n''$ should be false. We get a contradiction since $n'' + 0 = n''$ is a counterexample of $x + 0 = x$, smaller than $n' + 0 = n'$.

2.3 Term-based induction principles

Most of the term-based induction principles promote *eager* induction, an approach that defines the IHs (sometimes long) before their use by means of (explicit) *induction schemas* [2] usually resulted from the *recursion analysis* of recursively defined functions. Given a formula, an induction schema firstly identifies a subset of its variables to be instantiated, called *induction variables*, then defines the IHs as well as the induction conclusions as instances of the formula. The IHs associated to an induction conclusion are explicitly added to its conditions and are expected to participate in further developments of the proof. Some variables from the terms instantiating the induction variables are shared between the induction conclusions and their associated IHs.

A well-known example of 'induction schema'-based induction principle that fits well for constructor-based sorted specifications is the *structural induction* [38], which generalizes the Peano and mathematical inductions.

Structural induction. Let ϕ be a formula to be checked for the elements of a sort S. If $\forall f : S_1, \ldots, S_n \rightarrow S \in \mathcal{C}, \forall x_1, \ \ldots, \ \forall x_n, \ \{\phi(x_{i_1}) \cup \ldots \cup \phi(x_{i_k})\} \models_{\mathcal{M}} \phi(f(x_1, \ldots, x_n))$ then $\forall p \in S, \models_{\mathcal{M}} \phi(p)$, where the variables $x_{i_1}, \ldots, x_{i_k}$ are those variables among $x_1, \ldots, x_n$ that have the sort S.

Soundness proof. The structural induction principle exploits the fact that x is structurally smaller than $f(\ldots, x, \ldots)$. The term-based induction principle is applied further. □

In turn, it is generalized by the *'cover set' induction* [58] which is inspired from the idea of [10] according to which the induction schema is built from the recursive definition of the function appearing in the conjecture to be proved. The cover-set induction principle assumes that a sort is characterized, or covered, by a finite set of terms called (term) *cover-set*. The cover-set notion can be generalized to a set of term vectors that cover a product of sorts $S_1 \times S_2 \times \ldots$. Formally, $\{\overline{t_1}, \ldots, \overline{t_n}\}$ is a cover set of the product of sorts $\mathcal{E}$ if, for any formula ϕ, whenever $\not\models_{\mathcal{M}} \phi(\overline{u})$, for some term vector $\overline{u} \in \mathcal{E}$, it exists $j \in [1..m]$ and a substitution σ such that $\overline{t_j}\sigma \equiv \overline{u}$, where $\equiv$ is the (syntactical) identity relation.

Cover-set induction. Let Ψ be a non-empty cover set $\{\overline{t_1}, \ldots, \overline{t_m}\}$ of the product of sorts $\mathcal{E}$ and ϕ a formula to be checked for the elements of $\mathcal{E}$. If $\bigcup_{\overline{k} \in \mathcal{E}, \overline{k} <_t \overline{t}} \{\phi(\overline{k})\} \models_{\mathcal{M}} \phi(\overline{t})$, for any term vector $\overline{t} \in \Psi$, then $\forall \overline{p} \in \mathcal{E}, \models_{\mathcal{M}} \phi(\overline{p})$.

Soundness proof. By contradiction, assume that $\exists \overline{p'} \in \mathcal{E}$ such that $\models_{\mathcal{M}} \phi(\overline{p'})$ is false. We consider $\mathcal{E}' \subseteq \mathcal{E}$ defined as $\{\overline{p} \mid \overline{p} \in \mathcal{E}, \models_{\mathcal{M}} \phi(\overline{p}) \text{ is false}\}$. $\mathcal{E}'$ is not empty since $\overline{p'} \in \mathcal{E}'$. By the well-foundedness property of $<_t$ and Lemma 1, there is a minimal term vector $\overline{u} \in \mathcal{E}'$ such that $\not\models_{\mathcal{M}} \phi(\overline{u})$. By the definition of the term cover set, it exists $j \in [1..m]$ and a substitution σ such that $\overline{t_j}\sigma \equiv \overline{u}$. On the other hand, $\bigcup_{\overline{k} \in \mathcal{E}, \overline{k} <_t \overline{t_j}} \{\phi(\overline{k})\} \models_{\mathcal{M}} \phi(\overline{t_j})$. By the 'stability under substitutions' property of $<_t$, we have $\bigcup_{\overline{k} \in \mathcal{E}, \overline{k}\sigma <_t \overline{t_j}\sigma} \{\phi(\overline{k}\sigma)\} \models_{\mathcal{M}} \phi(\overline{t_j}\sigma)$. Since $\not\models_{\mathcal{M}} \phi(\overline{t_j}\sigma)$, it exists $\overline{k'} \in \mathcal{E}$ such that $\overline{k'} <_t \overline{u}$ and $\not\models_{\mathcal{M}} \phi(\overline{k'})$. Therefore, $\overline{k'} \in \mathcal{E}'$. Contradiction with the minimality of $\overline{u}$. □

The main shortcomings of the schemata-based approaches are related to the management of the IHs, when: i) the generated IHs do not contribute to the proof, and ii) the IHs that are required at some point of the proof are not yet generated or impossible to be defined with the 'recursion analysis' method. Its main advantage is the *local scope* of the induction ordering, at the level of the induction schema. This allows for flexibility in the ordering management during a proof; the ordering constraints are checked only when defining the induction schemas and, on the other hand, a different induction ordering may be used for each induction schema. Moreover, the schemata-based induction can be easily integrated into sequent-based deductive systems in terms of sound inference rules.

Another shortcoming is the treatment of mutual induction which cannot be directly performed because the induction conclusion and its attached IHs are instances of the *same* formula. However, several partial solutions have been proposed. Boyer and Moore [11] build the induction schema from a new function that calls the mutually defined functions according to the value of an extra argument. Sometimes, the conjecture should be strengthened or auxiliary lemmas about the mutually defined functions should be provided by the users. A more automatic solution has been provided by Kapur and Subramaniam [29] to deal with a class of mutually recursive functions that can transform the mutual recursion into simple recursion by unrolling the cover sets and expanding the function definitions. The multi-predicate induction schemas proposed by Boulton and Slind [9] have one predicate for each of the mutually recursive functions and avoid the need for expanding the functions into a single function. However, if the conjecture embeds different recursive function symbols the induction schemas have to be combined [10].

2.4 Formula-based induction principles

Inductionless induction, also known as *proof by consistency*, is the first proof method integrating formula-based induction. Proposed by Musser in [39], it uses the saturation-based Knuth-Bendix's completion algorithm [32] to prove inductive properties. The method can prove a set of equalities as consequences of a consistent set of equality axioms by i) adding them to the axioms, ii) orienting the new set of equalities into rewrite rules, and ii) showing their consistency if the completion algorithm saturates, i.e., until no new[2] equality is generated. As time went by, the method has been improved [5, 19, 21, 25, 26, 28, 35]. For an overview of inductionless induction, the reader may consult [16, 17].

By clearly separating the axioms from the formulas to be proved, the *implicit induction* inference systems are reductive procedures, many of them being rewrite-based and saturation-free. Initially, Reddy [44] proposed a proof method called *term rewriting induction*, implemented by an inference system which computes *formula cover-sets*. They are issued from the instantiation of some variables of a formula with a term cover-set associated to the product of their sorts defined in terms of *cover substitutions*. Each formula instance is reduced afterwards with rewrite rules from the axioms; in this way, whenever the formulas from the formula cover-set are valid, the covered formula is also valid. Bachmair [5] showed that formula cover sets are fundamental for the proofs by consistency. This is also true for the proofs by implicit induction. In the following, we refine formula cover sets as (general) cover sets and *strict* cover sets. Formally, $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is a set of cover substitutions of a formula $\phi(\overline{x})$ if the set of mapping term vectors built for each substitution from Σ is a term cover set of the (product) sort of $\overline{x}$. A set of formulas $\{\phi_1, \ldots, \phi_n\}$ is a formula cover set (resp. strict formula cover set) of a formula ϕ built on the set of cover substitution $\{\sigma_1, \ldots, \sigma_n\}$ if $\phi_i \models_{\mathcal{M}} \phi\sigma_i$ and $\phi_i \leq_f \phi\sigma_i$ (resp. $\phi_i <_f \phi\sigma_i$),

[2] w.r.t. some redundancy criteria.

for any $i \in [1..n]$.

The term rewriting method does not need to be *refutationally complete*, as required by [5], and the specifications may not be *ground confluent* in order to prove inductive properties.[3] Kounalis and Rusinowitch [34] went even further and proposed a completion-free induction technique based on *test-sets* [28].

To simplify the rest of the presentation, we will denote the ordering over formulas $<_f$ by $<$ and refer to (strict) formula cover sets as (strict) cover sets, if otherwise stated.

Term-rewriting induction. Let $\{\phi\sigma_1, \ldots, \phi\sigma_n\}$ be a set of instances of the formula $\forall \overline{x} \in \mathcal{E}, \phi(\overline{x})$ built from the cover-substitutions $\sigma_1, \ldots, \sigma_n$, respectively. If $\forall i \in [1..n], \bigcup_{\phi\theta < \phi\sigma_i} \phi\theta \models_{\mathcal{M}} \phi\sigma_i$ then $\forall \overline{x} \in \mathcal{E}, \models_{\mathcal{M}} \phi(\overline{x})$.

Soundness proof. By absurd, assume that $\forall i \in [1..n], \bigcup_{\phi\theta < \phi\sigma_i} \phi\theta \models_{\mathcal{M}} \phi\sigma_i$ but it exists $\overline{u} \in \mathcal{E}$ such that $\not\models_{\mathcal{M}} \phi(\overline{u})$. We consider $\mathcal{E}'$ defined as $\{\phi\theta \mid \not\models_{\mathcal{M}} \phi\theta\}$. $\mathcal{E}'$ is not empty since $\phi(\overline{u}) \in \mathcal{E}'$. Since $<$ is well-founded, there is a minimal instance $\phi\tau$ in $\mathcal{E}'$, by Lemma 1. Also, $\sigma_1, \ldots, \sigma_n$ are cover-substitutions for ϕ, so there exist $j \in [1..n]$ and a substitution ϵ such that $\phi\tau \equiv \phi\sigma_j\epsilon$ and $\not\models_{\mathcal{M}} \phi\sigma_j\epsilon$. Since $\bigcup_{\phi\theta < \phi\sigma_i} \phi\theta \models_{\mathcal{M}} \phi\sigma_i$ for all $i \in [1..n]$, this is also true for j, so $\bigcup_{\phi\theta < \phi\sigma_j} \phi\theta \models_{\mathcal{M}} \phi\sigma_j$. By the 'stability under substitutions' of $<$, we have $\bigcup_{\phi\theta\epsilon < \phi\sigma_j\epsilon} \phi\theta\epsilon \models_{\mathcal{M}} \phi\sigma_j\epsilon$. On the other hand, $\not\models_{\mathcal{M}} \phi\sigma_j\epsilon$, so there is a substitution θ' such that $\not\models_{\mathcal{M}} \phi\theta'\epsilon$ and $\phi\theta'\epsilon < \phi\sigma_j\epsilon (\equiv \phi\tau)$. Contradiction with the minimality of $\phi\tau$. □

The term-rewriting induction principle cannot directly perform mutual induction reasoning because all the involved formulas are instances of only one formula. It has been superseded by the implicit induction principle, as suggested in [34] and formally presented in [13].

Implicit induction. Let $\mathcal{E}$ be a set of formulas and assume that for any formula $\delta \in \mathcal{E}$, $\bigcup_{\gamma \in \mathcal{E}, \gamma < \delta} \{\gamma\} \models_{\mathcal{M}} \delta$. Also, let ϕ be a formula to be checked for a non-empty set $\mathcal{S}$ of term vectors. If $\forall \overline{p} \in \mathcal{S}, \phi(\overline{p}) \in \mathcal{E}$ then $\forall \overline{p} \in \mathcal{S}, \models_{\mathcal{M}} \phi(\overline{p})$.

Soundness proof. By the formula-based induction principle, any formula δ from $\mathcal{E}$ holds. Since $\forall \overline{p} \in \mathcal{S}, \phi(\overline{p}) \in \mathcal{E}$, then $\forall \overline{p} \in \mathcal{S}, \models_{\mathcal{M}} \phi(\overline{p})$. □

In practice, the implicit induction technique is applied on the set $\mathcal{E}$ of all instances of the formulas encountered in a proof derivation. The formula $\phi(\overline{p})$ is one of its initial conjectures and the ordering constraints from the relation $\forall \delta \in \mathcal{E}, \bigcup_{\gamma \in \mathcal{E}, \gamma < \delta} \{\gamma\} \models_{\mathcal{M}} \delta$ are guaranteed by *reductive* inference systems. They consist of *inference rules* representing transitions between *states* consisting of pairs of sets of formulas of the form (E, H), where E are *conjectures* and H are *premises*. By applying an inference rule, one of the conjectures, called *current conjecture*, is firstly transformed into a (potentially empty) set of new conjectures, then it may be added to the set of premises in order to participate to further transformations. A *derivation* is a successive application of inference rules. A *proof* of a set of formulas E^0 produced with the inference system I is a finite $n+1$-state derivation of the form $(E^0,\ \emptyset) \vdash_I (E^1,\ H^1) \vdash_I \ldots \vdash_I (\emptyset,\ H^n)$.

An inference system is *sound* if the minimal counterexamples are *persistent* in any derivation, i.e., whenever the current conjecture has a minimal counterexample, an equivalent one exists in the set of conjectures from a future state.

Theorem 1. *Let I be a sound inference system. For any proof $(E^0,\ \emptyset) \vdash_I \ldots \vdash_I (\emptyset,\ H^n)$, we have $\models_{\mathcal{M}} E^0$.*

[3] However, these properties are required to *refute* conjectures.

Proof. By contradiction, assume that E^0 has a false formula, hence a counterexample. By Lemma 1, in the set $\mathcal{E}$ there exists a minimal counterexample ϕ. Since I is sound, this counterexample is persistent and should be among the ground instances of the conjectures from the last state of the proof. Contradiction, since the proof finishes with an empty set of conjectures. □

A simple sound inference system is I^f:

GENERATE: $(E \cup \{\phi\},\ H) \vdash_{I^f} (E \cup \Psi,\ H \cup \{\phi\})$,
where Ψ is a strict cover set of ϕ.
SIMPLIFY: $(E \cup \{\phi\},\ H) \vdash_{I^f} (E \cup \Phi,\ H)$,
if $(E \cup \Phi \cup H)_{\leq \phi} \models_{\mathcal{M}} \phi$.

By $\Psi_{\leq \psi}$ (resp. $\Psi_{< \psi}$) are denoted the instances of formulas from Ψ that are smaller than or equal to (resp. strictly smaller than) ψ. The GENERATE rule replaces the current conjecture with a strict cover set of it, then adds it to the set of premises. SIMPLIFY replaces ϕ from the state $(E \cup \{\phi\}, H)$ with Φ if ϕ is a consequence of the set of IHs $(E \cup \Phi \cup H)_{\leq \phi}$.

Lemma 2. *The premises from any I^f-derivation starting with an empty set of premises do not have minimal counterexamples.*

Proof. Let us notice that the premises from any derivation that starts with an empty set of premises are added exclusively by GENERATE. By contradiction, assume that GENERATE applies on a current conjecture ϕ that has a minimal counterexample $\phi\tau$. By definition, the strict cover set Ψ for ϕ has a formula γ that covers $\phi\tau$, i.e., there is a cover substitution σ and another substitution ϵ such that $\phi\sigma\epsilon \equiv \phi\tau$ and $\bigcup_{\gamma < \phi\sigma}\{\gamma\} \models_{\mathcal{M}} \phi\sigma$. Thanks to the 'stability under substitutions' property of $<$, we have $\bigcup_{\gamma\epsilon < \phi\sigma\epsilon}\{\gamma\epsilon\} \models_{\mathcal{M}} \phi\sigma\epsilon$. On the other hand, $\not\models_{\mathcal{M}} \phi\sigma\epsilon$. Therefore, there is a ground instance from Ψ which is a counterexample smaller than $\phi\tau$. Contradiction. □

Theorem 2 (Soundness of I^f)**.** *I^f is sound for any derivation starting with an empty set of premises.*

Proof. Assume that a conjecture ϕ has a minimal counterexample $\phi\tau$. GENERATE cannot be applied on ϕ, by Lemma 2. If SIMPLIFY is applied to ϕ in the state $(E \cup \{\phi\}, H)$, we have $(E \cup \Phi \cup H)_{\leq \phi\tau} \models_{\mathcal{M}} \phi\tau$. Again by Lemma 2, H cannot have minimal counterexamples. Therefore, a minimal counterexample equivalent to $\phi\tau$ should exist in the conjectures $E \cup \Phi$ from the next state. □

I^f is an inference system that abstracts the computation, hence it cannot be used in practice. Its main role is to capture the induction reasoning by defining the formulas that can be used as IHs during a proof. Its concrete implementations show *how* the current conjecture is transformed, by the means of adequate reasoning techniques that may use the IHs defined by the implemented (abstract) inference rules. In addition, any such concrete implementation of I^f is sound since I^f has been shown sound by Theorem 2. The equality $x + 0 = x$ from the running example can be proved with the inference system I^f_c that allows to reason on equalities containing natural variables:

GenNat (G): $(E \cup \{\phi\},\ H) \vdash_{I_c^f} (E \cup \{\phi_1, \phi_2\},\ H \cup \{\phi\})$,
where ϕ has a natural variable that is instantiated by 0
and $S(x')$ and x' is a fresh variable; ϕ_1, ϕ_2 are
the result of rewriting the instances of ϕ with axioms.
SimpEq (S): $(E \cup \{\phi\},\ H) \vdash_{I_c^f} (E \cup \Phi,\ H)$,
if either i) ϕ is a tautology; in this case Φ is empty;
or, ii) ϕ is rewritten to ψ with rewrite rules from
$Ax \cup (E \cup \Phi \cup H)_{\leq \phi}$; in this case, Φ is $\{\psi\}$.

GenNat builds a strict cover set of an equality integrating a natural variable by firstly replacing the variable by 0 and the successor of a fresh natural variable, then rewriting the two instances by the rewrite rules resulted from orienting the axioms from left to right whenever the lhs is greater than the rhs. The rewriting results are stored as new conjectures and the equality as a new premise. Therefore, it implements Generate. On the other hand, SimpEq implements Simplify. It either deletes the tautologies or performs rewrite operations based on the same ordering over equalities mentioned in the footnote of Subsection 2.2, and on the IHs defined by Simplify.

The equality $x+0 = x$ can be proved with I_c^f, as follows: $(\{\underline{x + 0 = x}\}, \emptyset) \vdash^G_{I_c^f} (\{\underline{0 = 0}, S(x'+0) = S(x')\}, \{x+0 = x\}) \vdash^S_{I_c^f} (\{\underline{S(x' + 0) = S(x')}\}, \{x+0 = x\}) \vdash^S_{I_c^f} (\{\underline{S(x') = S(x')}\}, \{x+0 = x\}) \vdash^S_{I_c^f} (\emptyset, \{x + 0 = x\})$. In the derivation, the current conjectures from every state are underlined. The induction reasoning is performed during the second last SimpEq application: the instance $x'+0 = x'$ of the premise $x+0 = x$ is applied as IH in order to reduce $S(x'+0) = S(x')$ to the identity $S(x') = S(x')$. Since I_c^f is an instance of I^f, it is sound, so $x + 0 = x$ is valid in the initial model of Ax, by Theorem 1.

We present another inference system, denoted by I_s^f, that instantiates I^f. It integrates the saturation-based inference rule ConjSup:

ConjSup (Cs): $(E \cup \{\phi\},\ H) \vdash_{I_s^f} (E \cup \cup_i^n \{\phi_i\},\ H \cup \{\phi\})$,
where $\phi_i = \phi[r_i]_p\sigma_i$ for any rewrite rule $l_i = r_i$ from
Ax such that $\sigma_i = mgu(\phi|p, l_i)$ and $\phi|p$, with $i \in [1..n]$

ConjSup is adapted from [17] to perform *conjecture superposition* on a set of axioms that is saturated under superposition and equality reasoning. It firstly chooses from an equality ϕ one of the non-variable subterms $\phi|p$ at a position p, then unifies it with the left hand sides of all rewrite rules from the axioms. Any time the unification process is successful, the subterm is replaced by the corresponding unification instance of the right-hand side of the rewrite rule and the resulted equality becomes a new conjecture. If the set of new conjectures is not empty, the current conjecture is saved as premise. $\phi[r_i]_p$ indicates that ϕ has the subterm r_i at position p. In [48,49], it has been shown that $\cup_i^n\{\phi_i\}$ from similar superposition-based inference rules is a strict cover set of ϕ, as it is the case for ConjSup.

The I_s^f-proof of $x + 0 = x$ starts by applying the rule ConjSup on the subterm $x + 0$ of $x + 0 = x$. Since $x + 0$ unifies with the left-hand sides of the two axioms defining '+', the new conjectures are $0 = 0$ and $S(x' + 0) = S(x')$, where x' is a fresh variable. $x + 0 = x$ is added to the premises, to finally obtain a result similar to that of GenNat in the precedent I_c^f-proof. The rest of the proof can be successfully done as for the I_s^f-proof if I_s^f integrates a rule like SimpEq.

Different sound abstract reductive systems exist in the literature. I^f is very similar to the Implicit Induction procedure from [13], which is a generalization of the *hierarchical induction* procedure from [44] and of the inductive procedures for conditional equalities from [7, 12, 34]. A very general inference system was proposed in [47], based on the notion of *contextual* cover set, that generalizes those of cover set and strict cover set. It is conducted by a methodology to build sound implicit induction procedures using the compositional properties of contextual cover sets. The methodology also allowed to represent saturation-based inference systems as instances of the general inference system [48, 49]. It witnesses that the implicit induction and saturation-based procedures share the same logic. This is not surprising since Bachmair [5] and Reddy [44] already shown that the set of critical pairs generated by completion can build (strict) cover sets.

When dealing with equality reasoning, the reductive constraints between the current and new conjectures can be implicitly satisfied by the reductive inference rules if the equational specifications are represented in terms of rewrite systems and the IHs are orientable, as for the inductionless induction methods. Various solutions have been proposed to partially weaken the constraints related to IHs. In [50], the proposed method allows for *relaxed* rewriting [7] to deal with unorientable IHs by integrating explicit induction schemas when building cover sets. It covers the *term* [44], *ordered* [20], *enhanced* and *incremental* [1] rewriting induction procedures.

The formula-based induction reasoning can involve instances of *different* formulas, which makes easy the management of mutual induction [51]. As shown below, any term-based induction principle can be represented as formula-based one but the instantiation constraints are so strong that the ability to perform mutual induction is lost. This instantiation result argues a certain resemblance between (parts of) implicit and conventional induction proofs, firstly advocated in Musser's paper [39]. For example, the I^f-proof of the running example is very similar to the term-based induction version; as for an explicit induction schema, GENERATE instantiates variables and adds the current conjecture in the set of premises, but the resemblance stops here. [46] and [14] include similar instantiation results. Also, the formula-based induction procedures can perform *lazy* induction such that the IHs are provided by request. For example, during any of the proofs of $x + 0 = x$ with formula-based induction methods, the smaller instances of previous conjectures to be applied as IH are needed to be known only at the moment of their application.

Theorem 3. *Any term-based induction principle can be represented as a formula-based induction principle.*

Proof. Let us consider a term-based induction principle that proves the validity of a formula ϕ for all term vectors from a non-empty set $\mathcal{E}$, i.e., if for any term vector $\overline{m} \in \mathcal{E}$, $\bigcup_{\overline{k} \in \mathcal{E}, \overline{k} <_t \overline{m}} \{\phi(\overline{k})\} \models_{\mathcal{M}} \phi(\overline{m})$ then $\forall \overline{p} \in \mathcal{E}, \models_{\mathcal{M}} \phi(\overline{p})$. Let $\mathcal{E}'$ be the set $\{\phi(\overline{p}) \mid \overline{p} \in \mathcal{E}\}$. The equivalent formula-based induction principle can be stated as: if for any formula $\phi(\overline{m}) \in \mathcal{E}'$, $\bigcup_{\phi(\overline{k}) \in \mathcal{E}', \overline{k} <_t \overline{m}} \{\phi(\overline{k})\} \models_{\mathcal{M}} \phi(\overline{m})$ then we have $\models_{\mathcal{M}} \phi(\overline{p})$, $\forall \phi(\overline{p}) \in \mathcal{E}'$, where the weight of the formula $\phi(\overline{k})$ is that of $\overline{k}$ in the term ordering. □

Corollary 1. *Any term-based induction proof can be justified with formula-based induction arguments.*

In some cases, the implicit induction proofs are more automatic than those based on conventional induction [8], in other cases the contrary happens [29]. Further analyses and comparisons have been conducted in [16, 22, 27, 30, 40, 56].

3 A New Formula-based Induction Method

The features of formula- and term-based induction proof techniques mutually complement each other. In the following, we propose a new induction proof method which preserves the advantages of conventional and implicit induction techniques. The heart of the method is the DRaCuLa strategy, designed for performing:

- 'Descente Infinie' / Noetherian formula-based induction,
- Rarefied ordering constraints by reductive-free induction,
- Customized term-based induction, and
- Lazy and mutual induction.

A DRaCuLa-based inference system consists of a set of inference rules representing transitions between sets of conjectures. We introduce the inference system D made of three non-reductive abstract inference rules:

DEDUCTION (D): $E \cup \{\phi\} \vdash_D E \cup \Phi$,
if $\Phi \models \phi$.
SPLIT (S): $E \cup \{\phi\} \vdash_D E \cup \Phi$,
if Φ is a cover set of ϕ with at least two elements.
INDUCTION (I): $E \cup \{\phi\} \vdash_D E \cup \Phi$,
if $\Phi \cup \Psi \models_{\mathcal{M}} \phi$ and Ψ is a non-empty set of checked IHs.

DEDUCTION (resp. SPLIT) replaces the current conjecture by new formulas for which it is a deductive consequence (resp. by one of its cover sets). In addition, SPLIT is intended to deal with variable instantiations during a proof derivation, hence the requirement for the cover set to have at least two elements. INDUCTION treats the case when IHs are needed to build the new conjectures, but it can be applied only when the IHs are checked according to the DRaCuLa strategy.

It is assumed that each conjecture ϕ has attached a *history* consisting of a list of conjecture instances represented as pairs (ϕ^i, σ^i) and involved in producing ϕ. Formally, it is denoted by $\xrightarrow[...(\phi^i,\sigma^i)(\phi^{i+1},\sigma^{i+1})...]{} \phi$, where the history is shown under the horizontal arrow pointing to ϕ. The time flows from left to right, for example ϕ^i was created before ϕ^{i+1}. The conjectures from the history of ϕ are its *ancestors*. Each instantiating substitution from the history is an *identity* substitution, excepting when SPLIT is applied. For this case, the cover substitution involved in producing ϕ is considered instead. The *offsprings* of a formula ϕ are all formulas from a derivation having ϕ in the history. In addition, ϕ has attached a set of IHs to be checked before INDUCTION can be applied.

The DRaCuLa proof strategy. When a proof starts, the history and the set of attached IHs for each conjecture from the set of initial conjectures E^0 are empty. The DRaCuLa strategy mingles proof development, as implemented by the procedure 'Develop' (see Algorithm 1), with IH checking performed by the function 'Check' (see Algorithm 2). Algorithm 1 applies D-inference rules one by one starting from E^0. The application of any INDUCTION rule $E \cup \{\phi\} \vdash_D^I E \cup \Phi$ attaches the set Ψ of IHs to ϕ and is delayed until all IHs from Ψ are checked. In this case, ϕ is in *stand-by* and no other rule rule can be applied to it as long as the set of attached

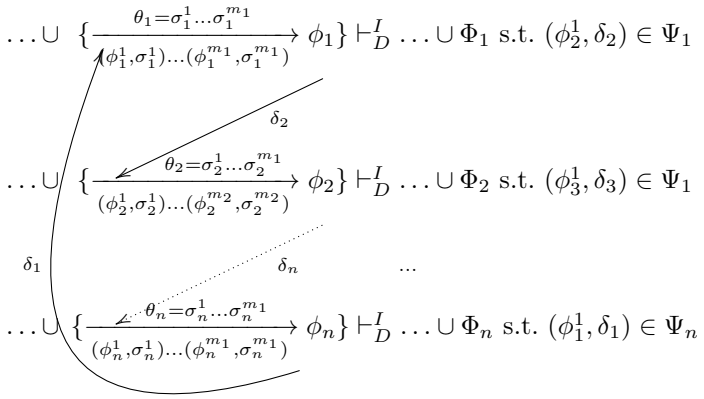


Figure 1: A cycle for checking n IHs.

IHs is not completely checked. Any IH is an instance (ϕ^1, δ), where ϕ^1 is a previous conjecture encountered in the derivation. Two cases may arise for successfully checking (ϕ^1, δ): i) if ϕ^1 was already proved, i.e., ϕ^1 has no offsprings in E, and ii) if (ϕ^1, δ) is part of a *cycle* built from ϕ and other conjectures in stand-by. Otherwise, the proof keeps developing other conjectures, hoping that the newly added conjectures build cycles that successfully check (ϕ^1, δ). The proof process successfully finishes when the set of conjectures becomes empty. Also, the proof development may be blocked if all conjectures from the current state are in stand-by. In practice, the deadlock can be avoided since other rules like Split and Deduction can be applied on the stand-by conjectures if the set of attached IHs is reinitialized. Finally, a proof may run infinitely if a crucial IH cannot be checked.

Definition 1 (n-cycle)**.** *Let us consider n conjectures $\phi_1, \ldots, \phi_n$ such that, for each $i \in [1..n]$, ϕ_i is explicitly represented with its history chunk involved in the cycle as $\xrightarrow[(\phi_i^1,\sigma_i^1)\ldots(\phi_i^{m_i},\sigma_i^{m_i})]{\theta_i=\sigma_i^1\ldots\sigma_i^{m_i}} \phi_i$ and has attached the IH $(\phi^1_{(i+1)\ mod\ n}, \delta_{(i+1)\ mod\ n})$. The n conjectures form a n-cycle if $\phi^1_{(i+1)\ mod\ n}\delta_{(i+1)\ mod\ n}$ is smaller than $\phi_i^1\theta_i$, for any $i \in [1..n]$.*

Figure 1 illustrates a n-cycle. The relation between the conjecture attaching an IH and the IH is graphically represented with a non-horizontal arrow. The *cumulative* substitution θ_i written above the horizontal arrow pointing to a conjecture ϕ_i allows to produce it from $\phi_i^1\theta_i$ using a Split-free derivation, for any $i \in [1..n]$. The main strength of the method is the fact that the induction reasoning involves only ordering constraints between instances of the conjectures starting the history chunks.

Building cycles. The cycle identification process may be costly if there are considered all the permutations built from subsets of the stand-by conjectures from the current state. Since each permutation represents a potential cycle, all the ancestors of the conjectures from the permutation may be tested to satisfy the ordering constraints. A more efficient alternative that avoids this combinatorial explosion problem is to build cycles incrementally and by need, as follows. Anytime a new IH (ϕ, δ) attached to a conjecture ϕ_0 is about to be checked and ϕ is not yet proved, the strategy for choosing the current conjecture privileges the offsprings of ϕ from the current state that already participate in the cycle. If the corresponding ordering

Algorithm 1 Develop(E): applies successively D-inference rules to build a proof of the conjectures from E

while E is not empty **do**
  IHS := $\{\psi \mid \exists \phi \in E$ that attached the unchecked IH $\psi\}$
  choose $\phi \in E$ with no unchecked IHs
  choose an inference rule to apply on ϕ
  if the set of IHs attached to ϕ is not empty **then**
    the chosen inference rule should be INDUCTION
  else
    if INDUCTION was chosen with the set Ψ of IHs **then**
      assign as unchecked and attach all IHs from Ψ to ϕ
      IHS := IHS $\cup$ Ψ
    end if
  end if
  if the chosen inference rule is INDUCTION **then**
    ok_IHs := $\emptyset$
    for all $(\phi_i^1, \delta_i) \in$ IHS **do**
      offsprings_ϕ_i^1 := $\{\phi' \mid \xrightarrow[hist]{} \phi' \in E$ and $\phi_i^1 \in hist\}$
      if offsprings_ϕ_i^1 is empty **then**
        ok_IHs := ok_IHs $\cup$ $\{(\phi_i^1, \delta_i)\}$ $\{\phi_i^1$ *was proved*$\}$
      end if
    end for
    ok_IHs := Check(IHS $\setminus$ ok_IHs, E) $\cup$ ok_IHs
    mark all IHs from ok_IHs as checked
  end if
  if all attached IHs to ϕ are checked **then**
    apply the chosen inference rule $E \cup \{\xrightarrow[hist]{} \phi\} \vdash_D E \cup \Phi$
    attach to each conjecture from Φ an empty set of IHs
    if inference rule is SPLIT **then**
      update any cover instance $\phi' \equiv (\phi, \sigma)$ to $\xrightarrow[hist;(\phi,\sigma)]{} \phi'$
    else
      update each $\phi' \in \Phi$ to $\xrightarrow[hist;(\phi,\sigma_{id})]{} \phi'$, where
      σ_{id} is the identity substitution instantiating all variables of ϕ.
    end if
    $E := (E \setminus \{\phi\}) \cup \Phi$
  end if
end while

Algorithm 2 Check(IHS, E): identifies cycles based on IHs from IHS and conjectures from E

Ensure: return all IHs from the identified cycles
  ok_IHs := $\emptyset$
  repeat
    find a non-empty list of n conjectures $\phi_1, \ldots, \phi_n$ from E, $\xrightarrow[\ldots(\phi_i^1,\sigma_i^1)\ldots(\phi_i^{m_1},\sigma_i^{m_1})]{} \phi_i$, $i \in [1..n]$
    if $(\phi_1^1, \delta_1) \in$ IHS and is attached to ϕ_n **and** $\phi_1^1\delta_1$ is smaller than $\phi_n^1\theta_n$, where θ_n is the cumulative substitution $\sigma_n^1 \ldots \sigma_n^{m_n}$ **then**
      if $n == 1$ **then**
        { 1-*cycle is found !*}
        ok_IHs := ok_IHs $\cup$ $\{(\phi_1^1, \delta_1)\}$
        IHS := IHS $\setminus$ $\{(\phi_1^1, \delta_1)\}$
      else
        if for each $i \in [1..n-1]$: $(\phi_{i+1}^1, \delta_{i+1}) \in$ IHS and is attached to ϕ_i, and $\phi_{i+1}^1\delta_{i+1}$ is smaller than $\phi_i^1\theta_i$, where θ_i is the cumulative substitution $\sigma_i^1 \ldots \sigma_i^{m_i}$ **then**
          {n-*cycle* ($n > 1$) *is found !*}
          ok_IHs := ok_IHs $\cup \cup_{i=1}^{n}$ $\{(\phi_i^1, \delta_i)\}$
          IHS := IHS $\setminus$ $\cup_{i=1}^{n}$ $\{(\phi_i^1, \delta_i)\}$
        end if
      end if
    end if
  until no cycle is found
  return ok_IHs

constraints are satisfied, the cycle is built. Otherwise, the ordering constraint related to the conjecture initiating the history chunk of ϕ_0 will be verified. If it is satisfied, a new history chunk starting with ϕ is added to the cycle. In this case, the proof of the offspring of ϕ continues to be developed until either it is proved, or a new IH required during the proof development is to be checked as previously. If the ordering constraint is not satisfied, Induction cannot be applied on ϕ and the proof of ϕ continues to be developed either by checking other IHs or by applying a rule other than Induction.

Definition 2 (D-proof)**.** *Any D-derivation built by Develop(E^0) and finishing with an empty set of conjectures is a D-proof of E^0, for any set of conjectures E^0.*

Theorem 4 (soundness of D)**.** *For any set of conjectures E^0, if there is a D-proof of E^0 then $\models_{\mathcal{M}} E^0$.*

Proof. By contradiction, assume that there is $\phi^0 \in E^0$ such that $\not\models_{\mathcal{M}} \phi^0$. Since Develop($E^0$) builds a derivation that finishes with an empty set of conjectures, there is a last step in the derivation when a false conjecture, denoted by ϕ', was processed. The applied rule is neither Deduction, nor Split because another false conjecture would be in the next step. So Induction has to be applied on ϕ'. The derivation should include at least one cycle checking IHs attached to false conjectures such that the new conjectures resulted from the application of the Induction rules from the cycle are true. Otherwise, the derivation does not perform inductive reasoning. More exactly, it can be transformed into a hierarchy of deductive proofs of conjectures from the proof of E^0, where the IHs are lemmas resulted from previously (deductively) proved conjectures, as follows. All the proofs that did not use IHs are at the bottom of the hierarchy, so they are true. The next upper level in the hierarchy consists of all proofs

using as lemmas the conjectures proved at the bottom level, so they are true, too. And so on, by stepping up in the hierarchy level by level, the current level proofs use as lemmas only conjectures proved at a lower level. The hierarchy is bounded since the proof of E^0 has a finite number of conjectures. It results that all conjectures from the proof of E^0, including ϕ^0, are true. Contradiction, since ϕ^0 is false.

A classical induction reasoning will be performed on the number of cycles checking IHs attached to false conjectures in the proof of E^0.

The base case: We assume that there is only one n-cycle checking IHs attached on false conjectures such that for each $i \in [1..n]$, ϕ_i is explicitly represented in the cycle as $\xrightarrow[(\phi_i^1,\sigma_i^1)...(\phi_i^{m_i},\sigma_i^{m_i})]{\theta_i=\sigma_i^1...\sigma_i^{m_i}} \phi_i$ and has attached the IH $(\phi^1_{(i+1) \bmod n}, \delta_{(i+1) \bmod n})$. Moreover, $\phi^1_{(i+1) \bmod n}{}^{\delta_{(i+1) \bmod n}}$ is smaller than $\phi_i^1\theta_i$, for any $i \in [1..n]$.

Any false instance of ϕ^0 should lead to one of the conjectures from the n-cycle, otherwise an extra cycle including false conjectures should exist in the proof. More exactly, for any counterexample $\phi^0\tau^0$, there is a conjecture ϕ from the n-cycle whose history is of the form $\xrightarrow[(\phi^0,\sigma^0)(\phi^1,\sigma^1).....(\phi^p,\sigma^p)]{} \phi$ and $\phi^0\tau^0$ is an instance of $\phi^0\sigma^0$. Moreover, any conjecture instance $\phi^i\theta^i$ is false, where θ^i is the cumulative substitution between ϕ^i and ϕ, for all $i \in [1..p]$. Otherwise, an extra cycle including false conjectures should exist in the proof, which leads to a contradiction.

W.l.o.g, we assume that ϕ is represented in the n-cycle by ϕ_n and the INDUCTION rule applied on ϕ_n is $E_n \cup \{\phi_n\} \vdash_D^I E_n \cup \Phi_n$ using the non-empty set Ψ_n of IHs such that $\Phi_n \cup \Psi_n \models_{\mathcal{M}} \phi_n$ and $\not\models_{\mathcal{M}} \phi_n$. Φ_n is valid, otherwise there is an extra cycle including false conjectures which is applied in the proof of Φ_n. Since Φ_n is valid and $\Phi_n \cup \Psi_n \models_{\mathcal{M}} \phi_n$, there is a false IH in Ψ_n. We can show that the IHs from Ψ_n, different from (ϕ_1^1, δ_1), are true. More exactly, if (ψ, δ) is such an IH, it cannot be proved only with deductive reasoning, so there should exist a new cycle checking ψ. This cycle cannot have false conjectures, so it includes neither ϕ_n nor any false offsprings of ψ. Therefore, ψ is true, which also holds for $\psi\delta$. We conclude that (ϕ_1^1, δ_1) is the only false IH attached to ϕ_n.

The history chunk of ϕ_n is represented in the n-cycle as $\xrightarrow[(\phi_n^1,\sigma_n^1)...(\phi_n^{m_n},\sigma_n^{m_n})]{\theta_n=\sigma_n^1...\sigma_n^{m_n}} \phi_n$. Since $\phi_n^1\theta_n$ is false, let $\phi_n^1\theta_n\tau$ be a minimal counterexample of it. Moreover, $\phi_n\tau$ is a counterexample because no cycle including false conjectures can be built on the path leading from $\phi_n^1\sigma_n^1$ to ϕ_n. Thanks to the 'stability under substitutions' property of the induction ordering, we deduce that $\phi_1^1\delta_1\tau$ is false and smaller than $\phi_n^1\theta_n\tau$. For similar reasons, it cannot be proved outside the n-cycle, so it is an instance of $\phi_1^1\theta_1$. Let this false instance be $\phi_1^1\theta_1\tau_1$ such that $\phi_1^1\theta_1\tau_1 \equiv \phi_1^1\delta_1\tau$. A similar reasoning is performed on $\phi_1^1\theta_1\tau_1$ as for $\phi_n^1\theta_n\tau$ to show that there is a false instance $\phi_2^1\theta_2\tau_2 \equiv \phi_2^1\delta_2\tau_1$ which is smaller than $\phi_1^1\theta_1\tau_1$. And so on, there is a false instance $\phi_n^1\theta_n\tau_n \equiv \phi_n^1\delta_n\tau_{n-1}$ which is smaller than $\phi_{n-1}^1\theta_{n-1}\tau_{n-1}$. By the transitivity of the ordering, we have that $\phi_n^1\theta_n\tau_n$ is smaller than $\phi_n^1\theta_n\tau$. Moreover, it is false, so contradiction with the minimality assumption of $\phi_n^1\theta_n\tau$.

The step case: We assume that Develop(E^0) has produced a proof having m (> 1) cycles including false conjectures. By induction hypothesis, any proof using a smaller number of cycles with false conjectures is sound.

We follow a reasoning similar to that employed for the base case. We will focus on the last generated cycle from the proof of E^0 having false conjectures. Assuming that it is built from n (> 1) conjectures, each ϕ_i with $i \in [1..n]$ is explicitly represented as $\xrightarrow[(\phi_i^1,\sigma_i^1)...(\phi_i^{m_i},\sigma_i^{m_i})]{\theta_i=\sigma_i^1...\sigma_i^{m_i}} \phi_i$

and has attached the IH $(\phi^1_{(i+1) \bmod n}, \delta_{(i+1) \bmod n})$. Moreover, $\phi^1_{(i+1) \bmod n}\delta_{(i+1) \bmod n}$ is smaller than $\phi^1_i\theta_i$, for any $i \in [1..n]$.

It can be noticed that any false instance of ϕ^0 should lead to one of the conjectures from the n-cycle, otherwise the proof of such a false instance should have 'less than m' cycles having false conjectures since the last cycle is not included in the proof. By the induction hypothesis, the proof is sound, so contradiction with the assumption that the instance of ϕ^0 is false. Therefore, for any counterexample $\phi^0\tau^0$ that is instance of $\phi^0\sigma^0$, there is a conjecture ϕ from the n-cycle whose history is of the form $\xrightarrow[(\phi^0,\sigma^0)(\phi^1,\sigma^1).....(\phi^p,\sigma^p)]{} \phi$. In addition, any conjecture instance $\phi^i\theta^i$ is false, where θ^i is the cumulative substitution between ϕ^i and ϕ, for all $i \in [1..p]$. Otherwise, $\phi^0\tau^0$ can be proved with 'less than m' cycles having false conjectures, so contradiction.

Again, we assume that ϕ is represented in the n-cycle by ϕ_n and the INDUCTION rule applied on ϕ_n is $E_n \cup \{\phi_n\} \vdash^I_D E_n \cup \Phi_n$ using the non-empty set Ψ_n of IHs such that $\Phi_n \cup \Psi_n \models_{\mathcal{M}} \phi_n$ and $\not\models_{\mathcal{M}} \phi_n$. Φ_n is valid, otherwise there is an extra cycle including false conjectures which is applied in the proof of Φ_n which contradicts the assumption that the n-cycle is the last generated one in the proof of E^0. Since Φ_n is valid and $\Phi_n \cup \Psi_n \models_{\mathcal{M}} \phi_n$, there is a false IH in Ψ_n. The IHs from Ψ_n, excepting (ϕ^1_1, δ_1), are true. More exactly, if (ψ, δ) is such an IH, it has to be proved with 'less than m' cycles having false conjectures since the n-cycle cannot be included in the proof, i.e., either (ψ, δ) is checked by a cycle including ϕ_n and generated before the n-cycle, or ψ was proved before generating the n-cycle. For the last case, ψ is true by induction hypothesis, which also holds for $\psi\delta$. We conclude that (ϕ^1_1, δ_1) is the only false IH attached to ϕ_n.

As for the base case, we represent the history chunk of ϕ_n from the n-cycle as $\xrightarrow[(\phi^1_n,\sigma^1_n)...(\phi^{m_n}_n,\sigma^{m_n}_n)]{\theta_n=\sigma^1_n...\sigma^{m_n}_n} \phi_n$. Since $\phi^1_n\theta_n$ is false, we consider $\phi^1_n\theta_n\tau$ as being a minimal counterexample of it. In addition, $\phi_n\tau$ should be a counterexample, otherwise $\phi^1_n\theta_n\tau$ can be proved with 'less than m' cycles having false conjectures. Thanks to the 'stability under substitutions' property of the induction ordering, we deduce that $\phi^1_1\delta_1\tau$ is false and smaller than $\phi^1_n\theta_n\tau$. It should be an instance of $\phi^1_1\theta_1$, otherwise it can be proved outside the n-cycle using 'less than m' cycles including false conjectures. We denote this false instance by $\phi^1_1\theta_1\tau_1$, hence $\phi^1_1\theta_1\tau_1 \equiv \phi^1_1\delta_1\tau$. A similar reasoning is performed on $\phi^1_1\theta_1\tau_1$ as for $\phi^1_n\theta_n\tau$ to show that there is a false instance $\phi^1_2\theta_2\tau_2 \equiv \phi^1_2\delta_2\tau_1$ which is smaller than $\phi^1_1\theta_1\tau_1$. And so on, there is a false instance $\phi^1_n\theta_n\tau_n \equiv \phi^1_n\delta_n\tau_{n-1}$ which is smaller than $\phi^1_{n-1}\theta_{n-1}\tau_{n-1}$. By the transitivity of the ordering, we have that $\phi^1_n\theta_n\tau_n$ is smaller than $\phi^1_n\theta_n\tau$. Moreover, it is false, so contradiction with the minimality assumption of $\phi^1_n\theta_n\tau$.

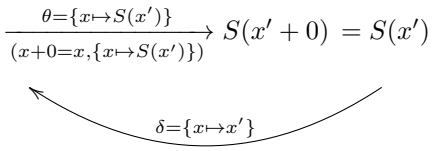
□

The DRaCuLa-based n-cycles from the D-proofs, for short *D-cycles* when n is not relevant, need less ordering constraints than the *reductive* cycles encountered in reductive induction derivations, for example the implicit induction and 'cyclic' proofs. More exactly, a reductive cycle requires that, for any history chunk $\xrightarrow[(\phi^0,\sigma^0)(\phi^1,\sigma^1)...(\phi^m,\sigma^m)]{} \phi$ with $m > 0$, $\phi^i\sigma^i$ to be greater than or (sometimes) equal to ϕ_{i+1}, for all $i \in [0..m-1]$. Moreover, $\phi^m\sigma^m$ should be greater than (or equal to) ϕ. As for the D-cycles, the IHs are instances of previous conjectures starting a history chunk, but they are required to be smaller than or equal to the conjecture they are applied to.

Lemma 3. *Any reductive n-cycle with an average of m conjectures per history chunk should satisfy $(m+1) \times n$ ordering constraints.*

Proof. There are $n \times m$ ordering constraints concerning the conjectures from the history chunks, and n ordering constraints related to the IHs. □

For example, there is only one reductive cycle in the I_c^f-proof of $x + 0 = x$:



The two associated ordering constraints are: $S(x')+0 = S(x')$ should be greater than $S(x'+0) = S(x')$ which should be greater than $x' + 0 = x'$.

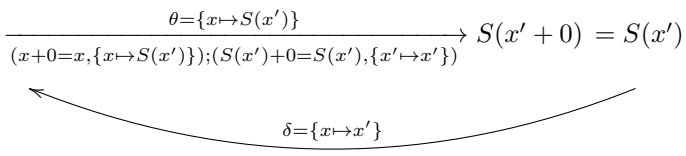
Lemma 4. *Any DRaCuLa-based n-cycle should satisfy n ordering constraints.*

Proof. By the construction of D-cycles. □

One possible D-proof of $x + 0 = x$, similar to that produced with the reductive I_c^b system, can be built with the concrete inference system D_c:

DedNat (D_c): $E \cup \{\phi\} \vdash_{D_c} E \cup \Phi$,
if either i) ϕ is a tautology; in this case Φ is empty,
or ii) ϕ is rewritten by rewrite rules from Ax to ψ;
in this case Φ is $\{\psi\}$.
SplitNat (S_c): $E \cup \{\phi[x]\} \vdash_{D_c} E \cup \{\phi[0], \phi[S(x')]\}$,
where x' is a fresh natural variable.
IndNat (I_c): $E \cup \{\phi\} \vdash_{D_c} E \cup \Phi$,
if ϕ is rewritten with an IH that is checked by a D-cycle.

The generated proof is: $\{\underline{x + 0 = x}\} \vdash_{D_c}^{S_c} \{\underline{0 + 0 = 0},\ S(x') + 0 = S(x')\} \vdash_{D_c}^{D_c(2)} \{\underline{S(x') + 0 = S(x')}\} \vdash_{D_c}^{D_c} \{\underline{S(x' + 0) = S(x')}\} \vdash_{D_c}^{I_c} \{\underline{S(x') = S(x')}\} \vdash_{D_c}^{D_c} \emptyset$. $\vdash_{D_c}^{D_c(2)}$ means that DedNat firstly rewrites with the axioms defining '$+$', then deletes the resulted tautology. The IH from the IndNat step is checked by the D-cycle:



Even if the history chunk has more conjectures, there is only one ordering constraint to be satisfied, i.e., $x' + 0 = x'$ should be smaller than $S(x') + 0 = S(x')$. It can be easily noticed that DedNat (resp. SplitNat, resp. IndNat) implements Deduction (resp. Split, resp. Induction). Therefore D_c is sound since D is sound, by Theorem 4.

Lemma 5. *Any reductive n-cycle is a DRaCuLa-based n-cycle.*

Proof. Assume that a reductive n-cycle exists, represented as a non-empty list of n conjectures $\phi_1, \ldots, \phi_n$ of the form $\xrightarrow[\ldots(\phi_i^1,\sigma_i^1)\ldots(\phi_i^{m_1},\sigma_i^{m_1})]{\theta_i=\sigma_i^1\ldots\sigma_i^{m_1}} \phi_i$, $i \in [1..n]$. By construction, for each $i \in [1..n]$, the reductive constraints require that $\phi_i^k\sigma_i^k$ be greater than (and sometimes equal to) ϕ_i^{k+1}

($k \in [1..m_1 - 1]$) and $\phi_i^{m_1}\sigma_i^{m_1}$ be greater than (or equal to) ϕ_i. By instantiating each $\phi_i^k\sigma_i^k$ by $\sigma_i^{k+1} \ldots \sigma_i^{m_1}$ and due to the 'stability under substitution' of the ordering, it results the decreasing sequence $\phi_i^1\theta_i^1$, $\phi_i^2\theta_i^2$, ..., $\phi_i^{m_1}\theta_i^{m_1}$, ϕ_i, where θ_i^j is the cumulative substitution $\sigma_i^j \ldots \sigma_i^{m_1}$. By the transitivity of the ordering relation, $\phi_i^1\theta_i^1$ is greater than ϕ_i. Moreover, by construction, ϕ_i should be greater than or equal to the IH $\phi^1_{(i+1)\bmod n}\delta_{(i+1)\bmod n}$. Applying again the transitivity property of the ordering relation, $\phi_i^1\theta_i^1$ is greater than $\phi^1_{(i+1)\bmod n}\delta_{(i+1)\bmod n}$.

Therefore, the reductive n-cycle is a DRaCuLa-based n-cycle since $\phi_i^1\theta_i^1$ is greater than $\phi^1_{(i+1)\bmod n}\delta_{(i+1)\bmod n}$, forall $i \in [1..n]$. □

Theorem 5 (generalisation of reductive induction)**.** *Any reductive proof is a D-proof.*

Proof. Given a reductive proof, its reductive cycles can be represented as D-cycles, by Lemma 5. Moreover, the DRaCuLa strategy requires no ordering constraints for the proof parts outside the reductive cycles. □

The DRaCuLa-based proofs are more flexible in terms of induction orderings because the ordering constraints inside the D-cycles can be formulated with different induction orderings. This is not the case for the reductive proofs which are governed by only one global induction ordering that should satisfy all the reductive constraints. As a side-effect, the axioms and IHs involved in a proof may satisfy additional constraints. For example, the term ordering used by rewrite-based specifications to orient the axioms into rewrite rules and the induction ordering should be compatible. The specifications fitting for DRaCuLa-based reasoning no longer need them, those adapted for term-based induction reasoning [55] being good candidates.

Theorem 6 (generalisation of term-based induction)**.** *Any term-based induction proof can be customized to a '*1*-cycle'-based D-proof.*

Proof. According to the term-based induction principle, one can use $\phi^1(\overline{k})$ as IH in the proof of $\phi^1(\overline{m})$, as long as $\overline{k} <_t \overline{m}$. On the other hand, according to Theorem 3, the ordering constraint can be reformulated using a formula-based principle as: $\phi^1(\overline{k})$ should be smaller than $\phi^1(\overline{m})$, by considering the weight of $\overline{v}$ as being that for $\phi^1(\overline{v})$, for any term vector $\overline{v}$. Assuming that $\phi^1(\overline{k})$ is an IH attached to the offspring ϕ_f of $\phi^1(\overline{m})$, the induction principle can be schematized by the 1-cycle

$$\xrightarrow[(\phi^1\sigma)\ldots(\phi',\sigma')]{\theta=\sigma\ldots\sigma'} \phi_f, \qquad \phi_f \xrightarrow{\ \delta\ } \phi^1(\overline{k})$$

where $\phi^1(\overline{m})$ is (ϕ^1, θ) and $\phi^1(\overline{k})$ is (ϕ^1, δ). □

For example, the explicit induction proof of $x + 0 = x$ builds a 1-cycle similar to the D-cycle issued from the reductive I_c^b-proof, excepting that the induction ordering is defined over terms. Its customisation to a D-cycle is done by defining the weight of the equality $x + 0 = x$ as being the term x, for any natural x.

4 Other Examples

The following set of axioms mutually defines over the naturals the functions *even*, *odd* and their conditional versions *even*1 and *odd*1, respectively:

$$even(0) = True \tag{1}$$
$$even(S(0)) = False \tag{2}$$
$$even(S(S(x))) = even1(S(S(x)) + 0) \tag{3}$$

$$even1(0) = True \tag{4}$$
$$even1(S(0)) = False \tag{5}$$
$$odd(x) = False \Rightarrow even1(S(S(x))) = even(x) \tag{6}$$
$$odd(x) = True \Rightarrow even1(S(S(x))) = False \tag{7}$$

$$odd(0) = False \tag{8}$$
$$odd(S(0)) = True \tag{9}$$
$$odd(S(S(x))) = odd1(S(S(x)) + 0) \tag{10}$$

$$odd1(0) = False \tag{11}$$
$$odd1(S(0)) = True \tag{12}$$
$$even(x) = True \Rightarrow odd1(S(S(x))) = odd(x) \tag{13}$$
$$even(x) = False \Rightarrow odd1(S(S(x))) = True \tag{14}$$

It can be noticed that all defined functions are terminating, sufficiently complete and consistent.

The conjectures to be proved are: $\phi_1^1 : even(x + x) = True$ and $\phi_2^1 : odd(y + y) = False$, using induction reasoning to be performed w.r.t. the initial model of the axioms. We assume that the previous proved conjecture $x + 0 = x$ is available as lemma, as well as the lemma $x + S(y) = S(x + y)$ which can be similarly proved. The ordering over conditional equalities is the multiset extension of the rpo ordering based on the precedence $<_F$ and equivalence $\sim_F$ relations over the function symbols: $True <_F False <_F 0 <_F S <_F + <_F (even \sim_F odd \sim_F even1 \sim_F odd1)$. The cyclic proof can be done using the inference system D_c if it is extended with the following rule:

DedCase (D_c'): $E \cup \{\phi\} \vdash_{D_c} E \cup \Phi$,
if a case analysis is performed on ϕ with rewrite rules
from Ax; in this case, Φ is made of the rewriting results.

The rewrite rules used by DedCase are conditional, of the form $d_1 = True \Rightarrow l_1 = r_1$ and $d_2 = False \Rightarrow l_2 = r_2$ such that both d_1 and d_2, resp. l_1 and l_2, are equal modulo renaming using the same renaming substitution. The case analysis is done as follows: if ϕ is an equality of the form $u = v$ such that u can be matched by l_1 and l_2 with the substitutions σ_1 and σ_2, respectively, then Φ consists of the set $\{d_1\sigma_1 = True \Rightarrow r_1\sigma_1 = v, d_2\sigma_2 = False \Rightarrow r_2\sigma_2 = v\}$. A tautology is an equality of the form $t = t$, or a conditional equality of the form $\ldots \Rightarrow t = t$ or $e \Rightarrow e$, for any term t and equality e.

The initial state of the D_c-proof is $\{\phi_1^1, \phi_2^1\}$. SplitNat is applied on ϕ_1^1 to result $\phi_1' : even(0 + 0) = True$ and $\phi_1^2 : even(S(x') + S(x')) = True$. ϕ_1' is rewritten by the axioms to the tautology $\phi_1 : True = True$, then deleted by DedNat. ϕ_1^2 is rewritten by the lemmas

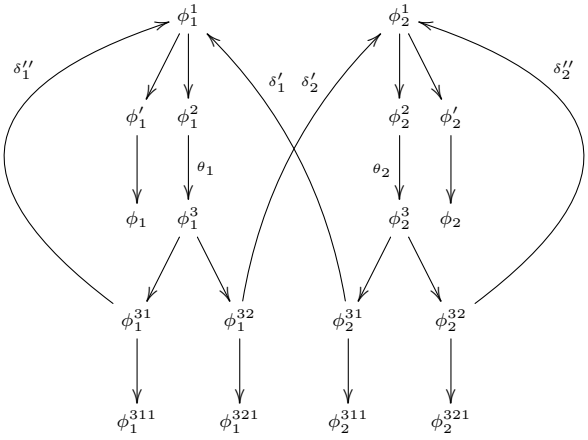


Figure 2: The skeleton of the D_c-proof.

$x + S(y) = S(x + y)$ and $x + 0 = x$ to $\phi_1^3 : even1(S(S(x' + x'))) = True$. By case analysis with DedCase on ϕ_1^3, it results $\phi_1^{31} : odd(x' + x') = False \Rightarrow even(x' + x') = True$ and $\phi_1^{32} : odd(x' + x') = True \Rightarrow False = True$. IndNat is applied on ϕ_1^{31} by rewriting with the IH (ϕ_1^1, δ_1''), where $\delta_1'' = \{x \mapsto x'\}$. The IH is checked by the 1-cycle represented by $\xrightarrow[(\phi_1^1,\{x\mapsto S(x')\})(\phi_1^2,id)(\phi_1^3,id)]{\theta_1=\{x\mapsto S(x')\}}$ ϕ_1^{31} since $\phi_1^1\delta_1''$ is smaller than $\phi_1^1\theta_1$. The rewriting operation produces the tautology $\phi_1^{311} : odd(x' + x') = True \Rightarrow True = True$ which is deleted by DedNat. Then, IndNat is expected to be applied on ϕ_1^{32} by rewriting this time with the IH (ϕ_2^1, δ_2'), where $\delta_2' = \{y \mapsto x'\}$. Hence, ϕ_1^{32} is put in stand-by and the proof of ϕ_2^1 starts by following similar steps as for ϕ_1^1. Firstly, SplitNat is applied to result $\phi_2' : odd(0 + 0) = False$ and $\phi_2^2 : odd(S(y') + S(y')) = False$. ϕ_2' is rewritten to the tautology $\phi_2 : False = False$, then deleted by DedNat. ϕ_2^2 is successively rewritten by the lemmas to $\phi_2^3 : odd1(y' + y') = False$ by DedNat, then by case analysis with DedCase to $\phi_2^{32} : even(y' + y') = True \Rightarrow odd(y' + y') = False$ and $\phi_2^{31} : even(y' + y') = False \Rightarrow True = False$. ϕ_2^{32} is further simplified to the tautology $\phi_2^{321} : even(y' + y') = True \Rightarrow False = False$ by IndNat with the IH (ϕ_2^1, δ_2''), where $\delta_2'' = \{y \mapsto y'\}$. The IH is checked by the 1-cycle with the history chunk $\xrightarrow[(\phi_2^1,\{y\mapsto S(y')\})(\phi_2^2,id)(\phi_2^3,id)]{\theta_2=\{y\mapsto S(y')\}}$ ϕ_2^{32} since $\phi_2^1\delta_2''$ is smaller than $\phi_2^1\theta_2$. IndNat can also be applied on ϕ_2^{31} by rewriting with the IH (ϕ_1^1, δ_1'), where $\delta_1' = \{x \mapsto y'\}$. The 2-cycle consisting of the history chunks $\xrightarrow[(\phi_1^1,\{x\mapsto S(x')\})(\phi_1^2,id)(\phi_1^3,id)]{\theta_1=\{x\mapsto S(x')\}}$ ϕ_1^{32} and $\xrightarrow[(\phi_2^1,\{y\mapsto S(y')\})(\phi_2^2,id)(\phi_2^3,id)]{\theta_2=\{y\mapsto S(y')\}}$ ϕ_2^{31} can check the two IHs since $\phi_2^1\delta_2'$ is smaller than $\phi_1^1\theta_1$ and $\phi_1^1\delta_1'$ is smaller than $\phi_2^1\theta_2$. The two IndNat operations are further applied to give the tautologies $\phi_1^{321} : False = True \Rightarrow False = True$ and $\phi_2^{311} : True = False \Rightarrow True = False$, which are finally deleted by DedNat. The cycles from the D_c-proof are highlighted in Figure 2.

Let us notice that the conjectures ϕ_1^1 and ϕ_2^1 cannot be proved by reductive reasoning since the axioms cannot be simultaneously oriented from left to right and transformed into rewrite rules, in particular (3) and (6), as well as (10), (14) and the axioms defining '+'. The D-proof cannot either be redone by term-based induction reasoning because of its 2-cycle.

#	conjecture	comparisons	IHs	cycles
1.	`firstat_timeat`	23	2	2
2.	`firstat_progat`	24	2	2
3.	`sorted_sorted`	5	0	0
4.	`sorted_insat1`	37	2	2
5.	`sorted_insin2`	45	2	2
6.	`sorted_e_two`	5	0	0
7.	`member_t_insin`	72	8	4
8.	`member_t_insat`	41	5	4
9.	`member_firstat`	37	3	3
10.	`timel_insat_t`	10	1	1
11.	`erl_insin`	11	1	1
12.	`erl_insat`	10	1	1
13.	`erl_prog`	38	2	2
14.	`time_progat_er`	20	1	1
15.	`timeat_tcrt`	16	1	1
16.	`timel_timeat_max`	43	1	1
17.	`null_listat`	17	2	2
18.	`null_listat1`	3	0	0
19.	`cons_insat`	4	1	1
20.	`cons_listat`	3	0	0
21.	`progat_timel_erl`	48	1	1
22.	`progat_insat`	156	4	2
23.	`progat_insat1`	63	3	2
24.	`timel_listupto`	7	0	0
25.	`sorted_listupto`	49	3	3
26.	`time_listat`	27	1	1
27.	`sorted_cons_listat`	62	2	2
28.	`null_wind2`	7	0	0
29.	`timel_insin1`	17	1	1
30.	`null_listupto1`	3	0	0
31.	`erl_cons`	11	0	0
32.	`no_time`	35	2	2
33.	`final`	29	2	2
	Total	978	54	46

Table 1: Statistics of the induction reasoning w.r.t. the ABR proofs.

In the following, we will show how the certification process of implicit induction proofs [53] can be improved by representing them as D-proofs. Even if the implicit induction proofs are generated automatically by inference systems that implicitly check the ordering constraints, the certification process should explicitly validate every single proof step. The number of ordering constraints, as indicated by Lemma 3, can be important. In [51], it has been shown that the validation of the ordering constraints for some proofs can last four times more than for the validation of the deductive reasoning. However, the validation time can be dramatically reduced if the implicit induction proofs are interpreted as D-proofs, as shown by Lemma 4. Table 1 gives some statistics about the proofs of a bunch of conjectures involved in the validation process [45]

of a conformance algorithm for the ABR protocol [43]. For example, the implicit induction proof of the conjecture `progat_insat` requires 152 reductive constraints and 4 applications of IHs. By inspecting the representation of the proof script as a D-proof, we have noticed that only two IHs are checked by 1-cycles, while the other two IHs do not need induction reasoning to be proved. Therefore, the validation process of the D-proof would require to check only 2 ordering constraints. The IHs not requiring induction reasoning can be proved in priority using a different proof strategy, then considered as lemmas during the rest of the proof.[4]

5 Conclusions and Future Work

We gave an overview of the induction proof methods for first-order logic and compared different induction principles and inference systems that abstract computation in order to distile the induction reasoning from the implementation details. Some examples of concrete implementations instantiating the abstract inference systems are presented, but most of the problems and challenges that face current implementations are not discussed here (see [24] for an overview).

We have proposed a new induction proof technique that captures the first-order induction reasoning by the means of non-reductive cycles which can be governed by different formula-based induction orderings. It has been shown enough powerful to subsume any term-based and reductive formula-based inductive proof methods by combining the best features of conventional and implicit induction proof techniques.

We have witnessed how the proposed technique can substantially diminish the number of reductive constraints during the certification process of implicit induction proofs. The DRaC-uLa strategy has been implemented into the Spike theorem prover but its potential for local inductive reasoning was not yet exploited. We intend to automatize the certification process of the D-proofs as it has been done for the implicit induction proofs issued by Spike. Recently, we have provided in [52] a procedure to translate a class of D-proofs into term-based induction proofs. This is the starting point for a longer-term project aiming to integrate our method into sequent-based systems.

Acknowledgments

We would like to thank the anonymous referees for the helpful remarks on previous versions of the paper.

References

[1] T. Aoto. Dealing with non-orientable equations in rewriting induction. In Frank Pfenning, editor, *RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2006.

[2] R. Aubin. Mechanizing structural induction. *Theor. Comput. Sci.*, 9:329–362, 1979.

[3] J. Avenhaus, U. Kühler, T. Schmidt-Samoa, and C.-P. Wirth. How to prove inductive theorems? QuodLibet! In Franz Baader, editor, *Proceedings of the 19th International Conference on Automated Deduction (CADE-19)*, number 2741 in Lecture Notes in Artificial Intelligence, pages 328–333. Springer, 2003.

[4] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[5] L. Bachmair. Proof by consistency in equational theories. *Logic in Computer Science, 1988. LICS '88., Proceedings of the Third Annual Symposium on*, pages 228–233, Jul 1988.

[4] The proof scripts can be accessed from `http://code.google.com/p/spike-prover/`

[6] A. Bouhoula, E. Kounalis, and M. Rusinowitch. Spike, an automatic theorem prover. In *Logic Programming and Automated Reasoning (LPAR)*, pages 460–462, 1992.

[7] A. Bouhoula, E. Kounalis, and M. Rusinowitch. Automated mathematical induction. *Journal of Logic and Computation*, 5(5):631–668, 1995.

[8] A. Bouhoula and M. Rusinowitch. Implicit induction in conditional theories. *Journal of Automated Reasoning*, 14(2):189–235, 1995.

[9] R. Boulton and K. Slind. Automatic derivation and application of induction schemes for mutually recursive functions. In J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. Pereira, Y. Sagiv, and P. Stuckey, editors, *Computational Logic — CL 2000*, volume 1861 of *Lecture Notes in Computer Science*, pages 629–643. Springer Berlin / Heidelberg, 2000.

[10] R. S. Boyer and J S. Moore. *A Computational Logic*. Academic Press, New York, NY, 1979.

[11] R. S. Boyer and J S. Moore. *A computational logic handbook*. Academic Press Professional, 1988.

[12] F. Bronsard and U. S. Reddy. Conditional rewriting in Focus. In *Conditional and Typed Rewriting Systems*, pages 1–13, 1991.

[13] F. Bronsard, U.S. Reddy, and R. Hasker. Induction using term orderings. In *Automated Deduction —CADE-12*, volume 814 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 1994.

[14] J. Brotherston and A. Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 2010.

[15] R. M. Burstall. Proving properties of programs by structural induction. *The Computer Journal*, 12:41–48, 1969.

[16] H. Comon. Inductionless induction. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 913–962. Elsevier and MIT Press, 2001.

[17] H. Comon and R. Nieuwenhuis. Induction= I-axiomatization+ first-order consistency. *Information and computation(Print)*, 159(1-2):151–186, 2000.

[18] J. Courant. Proof reconstruction. Research Report RR96-26, LIP, 1996. Preliminary version.

[19] N. Dershowitz. Applications of the Knuth-Bendix completion procedure. In *Seminaire d'Informatique Theorique*, pages 95–111, 1982.

[20] N. Dershowitz and U. S. Reddy. Deductive and inductive synthesis of equational programs. *Journal of Symbolic Computation*, 15(5/6):467–494, 1993.

[21] L. Fribourg. A strong restriction of the inductive completion procedure. In *ICALP (International Conference on Automata, Languages and Programming)*, volume 226 of *Lecture Notes in Computer Science*, pages 105–115, 1986. (Extended version in Journal of Symbolic Computation, Volume 8, Issue 3, September 1989, Pages 253-276).

[22] S. J. Garland and J. V. Guttag. Inductive methods for reasoning about abstract data types. *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 219–228, 1988.

[23] J. Goguen. How to prove algebraic inductive hypotheses without induction. In *5th Conference on Automated Deduction (CADE05)*, volume 87 of *Lecture Notes in Computer Science*, pages 356–373. Springer, 1980.

[24] B. Gramlich. Strategic issues, problems and challenges in inductive theorem proving. *Electronic Notes in Theoretical Computer Science*, 125(2):5–43, March 2005.

[25] G. Huet and J.M. Hullot. Proofs by induction in equational theories with constructors. Technical Report 0028, INRIA, 1980.

[26] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in equational theories without constructors. In Albert Meyer, editor, *Proceedings of the First Annual IEEE Symp. on Logic in Computer Science, LICS 1986*, pages 358–366. IEEE Computer Society Press, June 1986.

[27] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1):1 – 33, 1989. 4.1.5.

[28] D. Kapur, P. Narendran, and H. Zhang. Proof by induction using test sets. In *8th International*

Conference on Automated Deduction, volume 230 of *Lecture Notes Computer Science*, pages 99–117. Springer, 1986.

[29] D. Kapur and M. Subramaniam. Automating induction over mutually recursive functions. In *Algebraic Methodology and Software Technology*, volume 1101 of *Lecture Notes in Computer Science*, pages 117–131. Springer, 1996.

[30] D. Kapur and H. Zhang. Automating induction: Explicit vs. inductionless. *Proc. Third International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida, Jan*, pages 2–5, 1994.

[31] S. C. Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112:727–742, 1936.

[32] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297, 1970.

[33] E. Kounalis and M. Rusinowitch. A mechanization of conditional reasoning. In *First International Symposium on Artificial Intelligence and Mathematics*, 1990.

[34] E. Kounalis and M. Rusinowitch. Mechanizing inductive reasoning. In *Proceedings of the eighth National conference on Artificial intelligence - Volume 1*, AAAI'90, pages 240–245. AAAI Press, 1990.

[35] W. Küchlin. Inductive completion by ground proof transformation. In H. Ait-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures (Volume II): Rewriting Techniques*, pages 211–244. Academic Press, London, 1989.

[36] D. Lankford. Some remarks on inductionless induction. Technical Report MTP-11, Math. Dept., Louisiana Tech. Univ., Ruston, 1980.

[37] J. McCarthy. A basis for a mathematical theory of computation. In *Computer Programming and Formal Systems*, pages 33–70. North-Holland, 1963.

[38] J. McCarthy and J. Painter. Correctness of a compiler for arithmetic expressions. In *Mathematical Aspects of Computer Science*, pages 33–41. American Mathematical Society, 1967.

[39] D. R. Musser. On proving inductive properties of abstract data types. In *POPL*, pages 154–162, 1980.

[40] D. Naidich. On generic representation of implicit induction procedures. Technical Report CS-R9620, CWI, 1996.

[41] H. Poincaré. *Science et hypothèse*. Flammarion, 1902.

[42] M. Protzen. Lazy generation of induction hypotheses. *Automated Deduction —CADE-12*, pages 42–56, 1994.

[43] C. Rabadan and F. Klay. Un nouvel algorithme de contrôle de conformité pour la capacité de transfert 'Available Bit Rate'. Technical Report NT/CNET/5476, CNET, 1997.

[44] U.S. Reddy. Term Rewriting Induction. *Proceedings of the 10th International Conference on Automated Deduction*, pages 162–177, 1990.

[45] M. Rusinowitch, S. Stratulat, and F. Klay. Mechanical verification of an ideal incremental ABR conformance algorithm. *J. Autom. Reasoning*, 30(2):53–177, 2003.

[46] C. Sprenger and M. Dam. On the structure of inductive reasoning: Circular and tree-shaped proofs in the μ calculus. In A. Gordon, editor, *Foundations of Software Science and Computation Structures*, volume 2620 of *Lecture Notes in Computer Science*, pages 425–440. Springer Berlin / Heidelberg, 2003.

[47] S. Stratulat. A general framework to build contextual cover set induction provers. *J. Symb. Comput.*, 32(4):403–445, 2001.

[48] S. Stratulat. Automatic 'Descente Infinie' induction reasoning. In B. Beckert, editor, *TABLEAUX*, volume 3702 of *Lecture Notes in Artificial Intelligence*, pages 262–276. Springer, 2005.

[49] S. Stratulat. 'Descente Infinie' induction-based saturation procedures. In *SYNASC '07: Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific*

Computing, pages 17–24, Washington, DC, USA, 2007. IEEE Computer Society.

[50] S. Stratulat. Combining rewriting with Noetherian induction to reason on non-orientable equalities. In A. Voronkov, editor, *Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 351–365. Springer Berlin, 2008.

[51] S. Stratulat. Integrating implicit induction proofs into certified proof environments. In *Integrated Formal Methods*, volume 6396 of *Lecture Notes in Computer Science*, pages 320–335, 2010.

[52] S. Stratulat. Making explicit the implicit induction. submitted. Accessible at `http://lita.sciences.univ-metz.fr/~stratula/implicit2explicit.pdf`, 2012.

[53] S. Stratulat and V. Demange. Automated certification of implicit induction proofs. In *CPP'2011 (First International Conference on Certified Programs and Proofs)*, volume 7086 of *Lecture Notes Computer Science*, pages 37–53. Springer-Verlag, 2011.

[54] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[55] C. Walther. Mathematical induction. In Dov M. Gabbay, Christopher J. Hogger, J. A. Robinson, and Jörg H. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming (2)*, pages 127–228. Oxford University Press, 1994.

[56] C.-P. Wirth. History and future of implicit and inductionless induction: Beware the old jade and the zombie ! In *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, number 2605 in Lecture Notes in Artificial Intelligence, pages 192–203. Springer, 2005.

[57] C.-P. Wirth and B. Gramlich. On notions of inductive validity for first-order equational clauses. *Automated Deduction —CADE-12*, pages 162–176, 1994.

[58] H. Zhang, D. Kapur, and M. S. Krishnamoorthy. A mechanizable induction principle for equational specifications. In *Proceedings of the 9th International Conference on Automated Deduction*, pages 162–181, London, UK, 1988. Springer-Verlag.