Karlsruher Institut für Technologie

Karlsruhe Reports in Informatics 2012,2
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Rewriting Induction + Linear Arithmetic = Decision Procedure

Stephan Falke and Deepak Kapur

2012

KIT – University of the State of Baden-Wuerttemberg and National Research Center of the Helmholtz Association

Fakultät für **Informatik**

Rewriting Induction + Linear Arithmetic = Decision Procedure

Stephan Falke[1] and Deepak Kapur[2]

[1] Institute for Theoretical Computer Science, KIT, Germany
stephan.falke@kit.edu
[2] Dept. of Computer Science, University of New Mexico, USA
kapur@cs.unm.edu

Abstract. This paper presents new results on the decidability of inductive validity of conjectures. For this, a class of term rewrite systems (TRSs) with built-in linear integer arithmetic is introduced and it is shown how these TRSs can be used in the context of inductive theorem proving. The proof method developed for this couples (implicit) inductive reasoning with a decision procedure for the theory of linear integer arithmetic with (free) constructors. The effectiveness of the new decidability results on a large class of conjectures is demonstrated by an evaluation of the prototype implementation Sail2.

1 Introduction

Reasoning about the partial correctness of programs often requires proofs by induction, in particular for reasoning about recursive functions. There are two commonly used paradigms for inductive theorem proving: *explicit induction* and *implicit induction*. In explicit induction (see, e.g., [9, 39, 30, 11, 12, 22, 31, 38]), a concrete induction scheme is computed for each conjecture, and the subsequent reasoning is based on this induction scheme. Here, an induction scheme explicitly gives the base cases and the step cases, where a step case consists of an *obligation* and one or more *hypotheses*. In implicit induction (see, e.g., [33, 21, 26, 24, 16, 27, 34, 7, 1, 37]), no concrete induction scheme is constructed a priori. Instead, an induction scheme is implicitly constructed during the proof attempt.

Implicit induction is largely based on term rewriting. While ordinary term rewrite systems (TRSs) are a well-understood formalism for modelling algorithms, they lack in expressivity since they don't support built-in data structures such as integers. In the first part of this paper, an expressive class of TRSs (called $\mathbb{Z}$-*TRSs*) with built-in linear integer arithmetic is introduced. The semantics of integers can be utilized in $\mathbb{Z}$-TRSs in the form of linear integer arithmetic constraints (LIA-*constraints*). Then, an inductive proof method for $\mathbb{Z}$-TRSs is developed. This method couples implicit induction with a decision procedure for the theory LIAC, which combines linear integer arithmetic with the (free) constructors of the $\mathbb{Z}$-TRS.

While inductive proof methods can be automated, they do not provide a decision procedure since proof attempts may diverge, fail, or need intermediate

lemmas. In program verification, however, a decision procedure that can be used as a "black box" is preferable since an interactive use of inductive reasoning methods is typically only possible by trained experts. The goal of the second part of this paper is to derive conditions on $\mathbb{Z}$-TRSs and conjectures under which the proof method can be used as a decision procedure, i.e., will always produce a proof or disproof. These conditions are based on properties of the rewrite rules in a $\mathbb{Z}$-TRS that can be pre-computed during parsing. As we show experimentally in this paper, checking whether a conjecture satisfies the conditions is easily possible and requires much less time than attempting a proof or disproof.

Work on identifying conditions under which (explicit) inductive theorem proving with ordinary TRSs provides a decision procedure was initiated in [29] and later extended in [17, 18], also see [25]. These previous papers impose strong restrictions on both the TRSs and the conjectures. The functions defined by the TRS have to be given in such a way that any function f may only make recursive calls to the function f itself. Often, it is necessary to allow calls to other auxiliary functions or even mutually recursive definitions. Both of these possibilities are supported in this paper.

All of [29, 17, 18] impose the restriction that the conjectures contain a subterm of the form $f(x_1, \ldots, x_n)$ for a defined function f and pairwise distinct variables $x_1, \ldots, x_n$. This term is then chosen for the construction of the induction scheme upon which the proof is based. In this paper, much like in [15], this restriction is relaxed by making it possible to base the induction proof on a subterm where the arguments are not necessarily pairwise distinct variables. The result in this paper are however much more general than [15] since that paper did not yet consider $\mathbb{Z}$-TRSs. As a result, many conjectures which could not be handled previously can now be decided.

The integration of a decision procedures for the theory of linear integer arithmetic into inductive reasoning has been previously considered in [10, 28, 3], with the main focus on contextual rewriting which integrates rewriting with decision procedures. The proof method developed in this paper is in general incomparable to these methods, though, since instead of the more complex contextual rewriting without constraints, regular constrained rewriting is employed. This gives rise to an elegant and intuitive proof method. Inductive theorem proving for TRSs with constraints has been investigated in [8]. That method, however, does not support LIA-constraints and is thus incomparable to the method presented below. Another method for inductive theorem proving with constrained TRSs has been presented in [36], but that paper is only available in Japanese, unfortunately making it impossible for us to compare the inductive proof methods.

The results of this paper have been implemented in the prototype Sail2. The results of an evaluation on a large collection of examples are extremely encouraging. The checks for determining whether a conjecture falls in a decidable class of formulas can be efficiently implemented, taking less than 1% of the total time needed for a proof attempt. This suggests that the proposed method is viable as a decision procedure for inductive validity.

In summary, the contributions of the present paper are:

1. The introduction of a semantically expressive class of term rewrite system with linear integer arithmetic constraints.
2. The development of an inductive proof method for this class of rewrite systems that is based on the implicit induction paradigm coupled with a decision procedure for the theory that combines linear integer arithmetic and (free) constructors.
3. The identification of conditions on conjectures (both linear and nonlinear) and term rewrite systems under which this proof method is a decision procedure for inductive validity.
4. An implementation of the inductive proof method and a demonstration of the effectiveness of the approach on many examples.

The remainder of this paper is organized as follows: Section 2 introduces $\mathbb{Z}$-TRSs and Section 3 investigates properties of $\mathbb{Z}$-TRSs that are needed in the context of inductive theorem proving. The inductive proof method for $\mathbb{Z}$-TRSs is introduced in Section 4. Next, Section 5 investigates conditions under which the inductive proof methods can be used for deciding inductive validity. Section 6 discusses the implementation in Sail2 and presents an empirical evaluation. Finally, Section 7 concludes.

2 $\mathbb{Z}$-TRSs

This section introduces $\mathbb{Z}$-TRSs, a class of constrained TRSs that contains linear integer arithmetic as the built-in constraint theory. This class of TRSs can be seen as a simplified special case of the constrained equational rewrite systems (CERSs) from [14]. We assume familiarity with the notions and concepts from (many-sorted) term rewriting and refer to [5] for details.

For $\mathbb{Z}$-TRSs, built-in integers are modeled using the function symbols $\mathcal{F}_{\mathsf{LIA}} = \{\mathsf{0} : \to \mathtt{int},\ \mathsf{1} : \to \mathtt{int},\ - : \mathtt{int} \to \mathtt{int},\ + : \mathtt{int} \times \mathtt{int} \to \mathtt{int}\}$. Recall that $\mathbb{Z}$ is an Abelian group with unit $\mathsf{0}$ that is generated using the element $\mathsf{1}$. Integers thus satisfy the following properties $\mathcal{E}_{\mathsf{LIA}}$:

$$\begin{aligned} x + y &\approx y + x \\ x + (y + z) &\approx (x + y) + z \\ x + \mathsf{0} &\approx x \\ x + (-x) &\approx \mathsf{0} \end{aligned}$$

We use a simplified notation for terms built using $+$ and $-$, e.g., $x - y$ instead of $x + (-y)$, $\mathsf{2}$ instead of $\mathsf{1} + \mathsf{1}$, etc.

Properties of the built-in numbers are modeled using the predicate symbols $\mathsf{P} = \{>,\ \geq,\ \simeq\}$. The rewrite rules of $\mathbb{Z}$-TRSs then have constraints over these predicate symbols that guard when a rewrite step may be performed. To this end, an *atomic* LIA-*constraint* has the form $t_1\ P\ t_2$ for a predicate symbol $P \in \mathsf{P}$ and terms $t_1, t_2 \in \mathcal{T}(\mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ (where $\mathcal{V}$ is a set of variables). The set of LIA-*constraints* is the closure of the set of atomic LIA-constraints under $\top$ (truth), $\neg$ (negation), and $\wedge$ (conjunction). The Boolean connectives $\vee$, $\Rightarrow$, and $\Leftrightarrow$ can be

defined as usual. Also, LIA-constraints have the expected semantics. The main interest is in LIA-*satisfiability* (i.e., the constraint is true for some instantiation of its variables) and LIA-*validity* (i.e., the constraint is true for all instantiations of its variables). Notice that both of these properties are decidable.

For $\mathbb{Z}$-TRSs, $\mathcal{F}_{\mathsf{LIA}}$ is extended by a signature $\mathcal{F}$ of function symbols. In the following, *terms* denote members of $\mathcal{T}(\mathcal{F} \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ unless otherwise noted. A sequence $t_1, \ldots, t_n$ of terms is also denoted by t^*. Notions extend from terms to sequences of terms in the obvious way.

The left-hand sides of rules in a $\mathbb{Z}$-TRS may not contain arithmetical operations, i.e., they need to satisfy the following condition.

Definition 1 ($\mathbb{Z}$-Free Terms). *A term t is* $\mathbb{Z}$-free *iff it does not contain any occurrences of function symbols from $\mathcal{F}_{\mathsf{LIA}}$.*

Now the class of $\mathbb{Z}$-*TRSs* is defined as follows.

Definition 2 ($\mathbb{Z}$-TRSs, Defined Symbols, Constructors). *A* $\mathbb{Z}$-rule *is a rewrite rule of the form $l \to r[\![\varphi]\!]$ where l and r are terms with* $\mathrm{sort}(l) = \mathrm{sort}(r)$ *such that l is $\mathbb{Z}$-free and φ is a* LIA-*constraint. If $\varphi = \top$, then $l \to r$ can be written instead of $l \to r[\![\varphi]\!]$. A $\mathbb{Z}$-TRS is a finite set of $\mathbb{Z}$-rules. For a $\mathbb{Z}$-TRS $\mathcal{R}$, the set of* defined symbols *is $\mathcal{D}(\mathcal{R}) = \{f \mid f = \mathrm{root}(l)$ for some $l \to r[\![\varphi]\!] \in \mathcal{R}\}$. For $f \in \mathcal{D}(\mathcal{R})$, the set $\mathcal{R}(f) = \{l \to r[\![\varphi]\!] \in \mathcal{R} \mid \mathrm{root}(l) = f\}$ are the rules* defining *f. The set $\mathcal{C}(\mathcal{R}) = \mathcal{F} - \mathcal{D}(\mathcal{R})$ denotes the* constructors *of $\mathcal{R}$.*

Notice that the assumption that l is $\mathbb{Z}$-free is not severe in practice since an occurrence of a term $t \in \mathcal{T}(\mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ in the left-hand side can be replaced by a fresh variable x_t if $x_t \simeq t$ is added to the constraint of the rule.

It is assumed in the following that $\mathcal{C}(\mathcal{R})$ does not contain any function symbol with resulting sort `int`, i.e., no new constructors for $\mathbb{Z}$ are added. This is, of course, a natural assumption and no restriction.

Example 3. The following rules determine whether x is a divisor of y for two integers x, y:

$$\begin{array}{ll}
\mathsf{divides}(x,y) \to \mathsf{divides}(-x,y) & [\![x < 0 \wedge y \geq 0]\!] \\
\mathsf{divides}(x,y) \to \mathsf{divides}(x,-y) & [\![x \geq 0 \wedge y < 0]\!] \\
\mathsf{divides}(x,y) \to \mathsf{divides}(-x,-y) & [\![x < 0 \wedge y < 0]\!] \\
\mathsf{divides}(x,y) \to \mathsf{true} & [\![x \geq 0 \wedge y \simeq 0]\!] \\
\mathsf{divides}(x,y) \to \mathsf{true} & [\![x \simeq 0 \wedge y > 0]\!] \\
\mathsf{divides}(x,y) \to \mathsf{false} & [\![x > 0 \wedge y > 0 \wedge x > y]\!] \\
\mathsf{divides}(x,y) \to \mathsf{divides}(x,y-x) & [\![x > 0 \wedge y > 0 \wedge y \geq x]\!]
\end{array}$$

Then $\mathcal{D}(\mathcal{R}) = \{\mathsf{divides}\}$ and $\mathcal{C}(\mathcal{R}) = \{\mathsf{true}, \mathsf{false}\}$. △

The use of LIA-constraints makes it necessary to restrict the substitutions that may be used for rewriting with $\mathcal{R}$.

Definition 4 ($\mathbb{Z}$-Based Substitutions). *A substitution σ is* $\mathbb{Z}$-based *iff $\sigma(x) \in \mathcal{T}(\mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ for all variables x of sort* `int`.

The restriction that left-hand sides of rules are $\mathbb{Z}$-free allows for a simple definition of the rewrite relation of a $\mathbb{Z}$-TRS since LIA can be disregarded for matching. Notice that the matching substitution needs to be $\mathbb{Z}$-based in order to make sure that validity of the instantiated LIA-constraint can be decided.

Definition 5 (Rewrite Relation of a $\mathbb{Z}$-TRS). *Let $\mathcal{R}$ be a $\mathbb{Z}$-TRS and let s be a term. Then $s \to_{\mathcal{R},\mathbb{Z}} t$ iff there exist a constrained rewrite rule $l \to r[\![\varphi]\!] \in \mathcal{R}$, a position $p \in \mathcal{P}os(s)$, and a $\mathbb{Z}$-based substitution σ such that*

1. *$s|_p = l\sigma$,*
2. *$\varphi\sigma$ is LIA-valid, and*
3. *$t = s[r\sigma]_p$.*

Example 6. Using the $\mathbb{Z}$-TRS from Example 3, $\mathsf{divides}(2,-6) \to_{\mathcal{R},\mathbb{Z}} \mathsf{divides}(2,6)$ $\to_{\mathcal{R},\mathbb{Z}} \mathsf{divides}(2,4) \to_{\mathcal{R},\mathbb{Z}} \mathsf{divides}(2,2) \to_{\mathcal{R},\mathbb{Z}} \mathsf{divides}(2,0) \to_{\mathcal{R},\mathbb{Z}} \mathsf{true}$ using the first, seventh (three times), and fourth rule. △

3 Termination, Quasi-Reductivity, and Confluence

In the context of inductive theorem proving with ordinary TRSs it is typically required that the TRS satisfies certain properties. This section introduces these properties for $\mathbb{Z}$-TRSs. Additionally, it is discussed how these properties can be ensured for a class of $\mathbb{Z}$-TRSs that is sufficient for many practical purposes.

The first condition on a $\mathbb{Z}$-TRS is that it is terminating, i.e., that $\to_{\mathcal{R},\mathbb{Z}}$ is well-founded. While this is in general undecidable, the methods for proving termination of CERSs developed in [14] are applicable since $\mathbb{Z}$-TRSs can be seen as a restricted kind of CERSs. These methods are based on the dependency pair approach and have been implemented in the termination tool AProVE.

Example 7. Termination of the $\mathbb{Z}$-TRS from Example 3 can be easily established using AProVE. △

For the other properties that a $\mathbb{Z}$-TRS needs to satisfy, the following definition is needed.

Definition 8 (Constructor Ground Terms and Substitutions). *A ground term t is a* constructor ground term *if $t \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}})$. A ground substitution σ is a* constructor ground substitution *if $\sigma(x)$ is a constructor ground term for all variables x.*

In order to exclude degenerate cases, it is assumed that each sort has at least two distinct constructor ground terms. Notice that every constructor ground substitution is $\mathbb{Z}$-based since $\mathcal{C}(\mathcal{R})$ does not contain any function symbol with resulting sort `int`.

The second property of a $\mathbb{Z}$-TRS $\mathcal{R}$ that is needed for inductive theorem proving is that the defined functions in $\mathcal{D}(\mathcal{R})$ are total, i.e., result in a constructor ground term when applied to constructor ground terms.

Definition 9 (Quasi-Reductivity). *A $\mathbb{Z}$-TRS $\mathcal{R}$ is* quasi-reductive *iff every ground term of the form $f(t_1, \ldots, t_n)$ with $f \in \mathcal{D}(\mathcal{R})$ and constructor ground terms $t_1, \ldots, t_n$ is reducible by $\to_{\mathcal{R},\mathbb{Z}}$.*

Notice that quasi-reductivity is a special case of *sufficient completeness* [19]. The final property required of a $\mathbb{Z}$-TRS is *confluence*, where the definition of confluence is as follows.

Definition 10 (Confluence). *A $\mathbb{Z}$-TRS $\mathcal{R}$ is* confluent *iff $\leftarrow^*_{\mathcal{R},\mathbb{Z}} \circ \to^*_{\mathcal{R},\mathbb{Z}} \subseteq \to^*_{\mathcal{R},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R},\mathbb{Z}}$.*

Confluence implies the uniqueness of normal forms, whereas quasi-reductivity together with termination implies that each ground term can be reduced to a constructor ground term. Thus, if all three properties are satisfied, each ground term has a unique constructor ground term normal form.

Checking whether a $\mathbb{Z}$-TRS is quasi-reductive and confluent is a hard problem in general. Thus, a restricted class of $\mathbb{Z}$-TRSs is considered in the following. For this class, checking for quasi-reductivity is easily possible. Furthermore, $\mathbb{Z}$-TRSs from this class will always be confluent. First, it is required that the left-hand sides of rules are linear and constructor-based. This is important in order to check for quasi-reductivity. In order to ensure confluence, it is required that the rules are "disjoint" in the sense that at most one rule is applicable to each position in any term. Notice that two rules might have identical left-hand sides as long as the conjunction of the LIA-constraints of these rules is unsatisfiable.

Definition 11 (Orthogonal $\mathbb{Z}$-TRSs). *A $\mathbb{Z}$-TRS $\mathcal{R}$ is* orthogonal *iff*

1. *For all $l \to r[\![\varphi]\!] \in \mathcal{R}$, the term l is linear and has the form $f(l_1, \ldots, l_n)$ with $l_1, \ldots, l_n \in \mathcal{T}(\mathcal{C}(\mathcal{R}), \mathcal{V})$.*
2. *For any two rules $l_1 \to r_1[\![\varphi_1]\!], l_2 \to r_2[\![\varphi_2]\!]$, either $l_1 = l_2$[3] or l_1, l_2 are not unifiable after their variables have been renamed apart.*
3. *For any two non-identical rules $l_1 \to r_1[\![\varphi_1]\!], l_2 \to r_2[\![\varphi_2]\!]$ with $l_1 = l_2$, the constraint $\varphi_1 \wedge \varphi_2$ is* LIA*-unsatisfiable.*
4. *Whenever $l_1 \to r_1[\![\varphi_1]\!], \ldots, l_n \to r_n[\![\varphi_n]\!]$ are all rules with identical left-hand sides, then the constraint $\varphi_1 \vee \ldots \vee \varphi_n$ is* LIA*-valid.*

Example 12. The $\mathbb{Z}$-TRS from Example 3 is orthogonal since conditions *1* and *2* are obviously satisfied, the disjunction of the constraints of all rules is LIA-valid, and the conjunction of the constraints of any two distinct rules is LIA-unsatisfiable. △

Using conditions *1*, *2*, and *4* of this definition, the following decidability result can be obtained by reducing quasi-reductivity of orthogonal $\mathbb{Z}$-TRSs to quasi-reductivity of ordinary TRSs.

Theorem 13. *It is decidable whether an orthogonal $\mathbb{Z}$-TRS is quasi-reductive.*

[3] This could be relaxed by identifying terms that only differ by a variable renaming.

Somewhat surprisingly, orthogonal $\mathbb{Z}$-TRSs are always confluent, regardless of whether they are terminating or not. This result follows from conditions *1*, *2*, and *3* in Definition 11 since these conditions imply that orthogonal $\mathbb{Z}$-TRSs are a suitable generalization of orthogonal ordinary TRSs (which are also known to be confluent [35], regardless of whether they are terminating or not).

Theorem 14. *Every orthogonal $\mathbb{Z}$-TRS is confluent.*

4 Inductive Theorem Proving With $\mathbb{Z}$-TRSs

In the following, it is assumed that $\mathcal{R}$ is a terminating quasi-reductive orthogonal $\mathbb{Z}$-TRS, which implies that $\mathcal{R}$ is confluent.

The atomic conjectures in inductive theorem proving are equalities between terms. In this paper, a generalized form of these atomic conjectures is used that also incorporates a LIA-constraint.

Definition 15 (Atomic Conjectures). *An* atomic conjecture *has the form $s \equiv t[\![\varphi]\!]$ where s and t are terms with $\mathrm{sort}(s) = \mathrm{sort}(t)$ and and φ is a LIA-constraint. If $\varphi = \top$, then $s \equiv t$ can be written instead of $s \equiv t[\![\varphi]\!]$.*

Notice that atomic conjectures satisfy the same requirements as the rewrite rules in $\mathbb{Z}$-TRSs except that neither s nor t is required to be $\mathbb{Z}$-free. Intuitively, an atomic conjecture $s \equiv t[\![\varphi]\!]$ is true whenever $s\sigma$ and $t\sigma$ are "equal" up to the rules of $\mathcal{R}$ and the properties of $\mathcal{E}_{\mathsf{LIA}}$ for all constructor ground substitutions σ that make φ true.

Definition 16 (Inductive Theorems). *An atomic conjecture $s \equiv t[\![\varphi]\!]$ is an* inductive theorem *iff $s\sigma \leftrightarrow^*_{\mathcal{R} \cup \mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t\sigma$ for all constructor ground substitutions σ such that $\varphi\sigma$ is LIA-valid. A set of atomic conjectures is an* inductive theorem *iff all of its elements are inductive theorems.*

The inductive theorem proving method for $\mathbb{Z}$-TRSs developed in this paper is based on Reddy's *term rewriting induction* [34]. The presentation follows [1, 2]. The main idea of this method is to *expand* certain subterms of an atomic conjecture using narrowing with the rewrite rules of $\mathcal{R}$.

Definition 17 (Basic Terms). *A $\mathbb{Z}$-free term t is* basic *iff $t = f(t_1, \ldots, t_n)$ where $f \in \mathcal{D}(\mathcal{R})$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}(\mathcal{R}), \mathcal{V})$.*

Expansion of a basic subterm is now done as follows. Notice that the constraints of the atomic conjecture and the rewrite rule are combined and instantiated and that the unifier σ is always $\mathbb{Z}$-based due to the requirements on $\mathbb{Z}$-TRSs and atomic conjectures.

Definition 18 (Expd**).** *For an atomic conjecture $s \equiv t[\![\varphi]\!]$ and a basic term u such that $s = C[u]$, the set $\mathrm{Expd}_u(s, t, \varphi)$ is defined as*

$$\mathrm{Expd}_u(s,t,\varphi) = \{C[r]\sigma \equiv t\sigma[\![\varphi\sigma \wedge \psi\sigma]\!] \mid l \to r[\![\psi]\!] \in \mathcal{R},\ \sigma = \mathrm{mgu}(u,l) \text{ exists, and } \varphi\sigma \wedge \psi\sigma \text{ is } \mathsf{LIA}\text{-satisfiable} \}$$

Here, it has been assumed that the variables of $l \to r[\![\psi]\!]$ have been renamed to be disjoint from the variables of $s \equiv t[\![\varphi]\!]$.

Example 19. Consider the atomic conjecture $\mathsf{divides}(x, x) \equiv \mathsf{true}$ for the $\mathbb{Z}$-TRS from Example 3. For $s = \mathsf{divides}(x, x)$, $t = \mathsf{true}$, $u = s$, and $\varphi = \top$, $\mathrm{Expd}_u(s, t, \varphi) = \{\mathsf{divides}(-x, -x) \equiv \mathsf{true}\ [\![x < 0 \wedge x < 0]\!],\ \mathsf{true} \equiv \mathsf{true}\ [\![x \geq 0 \wedge x \simeq 0]\!],\ \mathsf{divides}(x, x - x) \equiv \mathsf{true}\ [\![x > 0 \wedge x > 0 \wedge x \geq x]\!]\}$. △

The following technical result relates the atomic conjectures in $\mathrm{Expd}_u(s, t, \varphi)$ to the atomic conjecture $s \equiv t[\![\varphi]\!]$ and the rules in $\mathcal{R}$. It is needed for the soundness proof of the inductive proof method (see Theorem 28 below).

Lemma 20. *Let $s \equiv t[\![\varphi]\!]$ be an atomic conjecture and let u be a basic term such that $s = C[u]$.*

1. *$s\sigma \to_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow_{\mathrm{Expd}_u(s,t,\varphi),\mathbb{Z}} t\sigma$ for all constructor ground substitutions σ such that $\varphi\sigma$ is* LIA*-valid.*
2. *If $v \leftrightarrow_{\mathrm{Expd}_u(s,t,\varphi),\mathbb{Z}} w$, then $v \leftrightarrow^*_{\mathcal{R}\cup\{s\equiv t[\![\varphi]\!]\},\mathbb{Z}} w$.*

The inductive proof method for $\mathbb{Z}$-TRSs is given by the inference system in Figure 1. Here, the notation $s \doteq t[\![\varphi]\!]$ is used to stand for one of $s \equiv t[\![\varphi]\!]$ and $t \equiv s[\![\varphi]\!]$. The inference rules operate on tuples $\langle E, H\rangle$, where E consists of atomic conjectures that are to be proven and H consists of atomic conjectures that have been oriented as rewrite rules. These rules constitute the *hypotheses* in a proof by induction. The goal of an inductive proof attempt is to obtain a tuple of the form $\langle \emptyset, H\rangle$ starting from the tuple $\langle E, \emptyset\rangle$. As shown below, this implies that E is an inductive theorem. On the other hand, if none of the inference rules is applicable to $\langle E', H'\rangle$ where $E' \neq \emptyset$, then the inductive proof attempt fails. Finally, an inductive proof attempt may also diverge (i.e., not terminate) or end in $\bot$. As shown in Theorem 30, the later constitutes a disproof of (at least) one of the atomic conjectures from E.

The inference rule Expand uses Definition 18 to expand a basic subterm of an atomic conjecture. Then, this atomic conjecture is oriented as a rewrite rule and added to the set H of hypotheses. Notice that this addition is only allowed if the $\mathbb{Z}$-TRS consisting of $\mathcal{R} \cup H$ and this newly obtained rule is terminating. This restriction is needed in order to obtain a sound inductive proof method.

The rule Simplify uses simplification with $\mathcal{R}$ and the hypotheses in H. For this, the constraint of the atomic conjecture that is to be simplified is taken into account by considering the following rewrite relation. It only differs from Definition 5 in condition *2*, which now requires that the instantiated constraint of the rewrite rule is valid under the assumption of the constraint of the atomic conjecture that is getting simplified.

Definition 21 (Rewrite Relation of a $\mathbb{Z}$-TRS on Constrained Terms). *Let $\mathcal{R}$ be a $\mathbb{Z}$-TRS, let s be a term, and let ψ be a* LIA*-constraint. Then $s[\![\psi]\!] \to_{\mathcal{R},\mathbb{Z}} t[\![\psi]\!]$ iff there exist a constrained rewrite rule $l \to r[\![\varphi]\!] \in \mathcal{R}$, a position $p \in \mathcal{P}os(s)$, and a $\mathbb{Z}$-based substitution σ such that*

Expand	$\dfrac{\langle E \uplus \{s \doteqdot t[\![\varphi]\!]\},\ H\rangle}{\langle E \cup \mathrm{Expd}_u(s,t,\varphi),\ H \cup \{s \to t[\![\varphi]\!]\}\rangle}$	if $\mathcal{R} \cup H \cup \{s \to t[\![\varphi]\!]\}$ terminates and s is $\mathbb{Z}$-free
Simplify	$\dfrac{\langle E \uplus \{s \doteqdot t[\![\varphi]\!]\},\ H\rangle}{\langle E \cup \{s' \doteqdot t[\![\varphi]\!]\},\ H\rangle}$	if $s[\![\varphi]\!] \to_{\mathcal{R}\cup H,\mathbb{Z}} s'[\![\varphi]\!]$
Case-Simplify	$\dfrac{\langle E \uplus \{s \doteqdot t[\![\varphi]\!]\},\ H\rangle}{\langle E \cup \{s' \doteqdot t[\![\varphi']\!] \mid s'[\![\varphi']\!] \in \mathrm{Case}_p(s,\varphi)\},\ H\rangle}$	
Delete	$\dfrac{\langle E \uplus \{s \doteqdot t[\![\varphi]\!]\},\ H\rangle}{\langle E,\ H\rangle}$	if $s \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t$ or φ is LIA-unsatisfiable
$\mathsf{Theory}_\top$	$\dfrac{\langle E \uplus \{s \doteqdot t[\![\varphi]\!]\},\ H\rangle}{\langle E,\ H\rangle}$	if s,t do not contain symbols from $\mathcal{D}(\mathcal{R})$ and $\varphi \Rightarrow s \simeq t$ is LIAC-valid
$\mathsf{Theory}_\bot$	$\dfrac{\langle E \uplus \{s \doteqdot t[\![\varphi]\!]\},\ H\rangle}{\bot}$	if s,t do not contain symbols from $\mathcal{D}(\mathcal{R})$ and $\varphi \Rightarrow s \simeq t$ is not LIAC-valid

Fig. 1. The inference system $\mathcal{I}$, where $s \doteqdot t[\![\varphi]\!]$ denotes one of $s \equiv t[\![\varphi]\!]$ and $t \equiv s[\![\varphi]\!]$.

1. $s|_p = l\sigma$,
2. $\psi \Rightarrow \varphi\sigma$ *is* LIA*-valid, and*
3. $t = s[r\sigma]_p$.

The inference rule Case-Simplify combines a case split with simplification using $\mathcal{R}$ (but not using H). It makes use of the following definition.

Definition 22 (Case)**.** *For a term s, a* LIA*-constraint φ, and a position $p \in \mathcal{P}os(s)$, the set* $\mathrm{Case}_p(s,\varphi)$ *is defined as*

$$\mathrm{Case}_p(s,\varphi) = \{s[r_i\sigma_i]_p[\![\varphi \wedge \psi_i\sigma_i]\!] \mid l_i \to r_i[\![\psi_i]\!] \in \mathcal{R}',\ s|_p = l_i\sigma_i, \text{ and } \varphi \wedge \psi_i\sigma_i \text{ is LIA-satisfiable } \}$$

Here, $\mathcal{R}' = \{l_1 \to r_1[\![\psi_1]\!], \ldots, l_n \to r_n[\![\psi_n]\!]\} \subseteq \mathcal{R}$ contains all rules whose left-hand side match $s|_p$. The construction can only be performed if $\sigma_1, \ldots, \sigma_n$ are $\mathbb{Z}$-based and $\varphi \Rightarrow \psi_1\sigma_1 \vee \ldots \vee \psi_n\sigma_n$ is LIA*-valid.*

Notice that the rule Case-Simplify can be used in order to simulate the rule Simplify in cases where Simplify uses a rule from $\mathcal{R}$ and not a hypothesis from H. The rule Delete removes trivial atomic conjectures, and the rules $\mathsf{Theory}_\top$ and $\mathsf{Theory}_\bot$ apply to atomic conjectures that do not contain any defined symbols and make use of a decision procedure for the theory LIAC that combines the linear theory of integers with the constructor symbols from $\mathcal{C}(\mathcal{R})$.

Definition 23 (LIAC)**.** *For a $\mathbb{Z}$-TRS $\mathcal{R}$, the theory* LIAC *has the form* $\mathsf{LIAC} = (\mathcal{F}_{\mathsf{LIAC}}, \mathsf{P}_{\mathsf{LIAC}}, \mathcal{M}_{\mathsf{LIAC}})$ *where*

1. $\mathcal{F}_{\mathsf{LIAC}} = \mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$

2. $\mathsf{P}_{\mathsf{LIAC}} = \{\simeq, \geq, >\}$[4]
3. $\mathcal{M}_{\mathsf{LIAC}} = (M, (f^{\mathsf{LIAC}})_{f \in \mathcal{F}_{\mathsf{LIAC}}}, (P^{\mathsf{LIAC}})_{P \in \mathsf{P}_{\mathsf{LIAC}}})$ *where* $M = \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathbb{Z})$*, the function symbols in* $\{0, 1, +, -\}$ *and predicate symbols in* $\mathsf{P}_{\mathsf{LIAC}}$ *are interpreted in the obvious way, and* $f^{\mathsf{LIAC}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ *for* $f \in \mathcal{C}(\mathcal{R})$*.*

LIAC-validity and LIAC-satisfiability are decidable and decision procedures have been implemented, for instance in the SMT-solver CVC3 [6].

Example 24. For two lists, $\mathsf{maxlist}(xs, ys)$ computes the list containing the pointwise maximum of these lists, stopping as soon as either list is empty. The function zip combines two lists into a list of pairs. Finally, fst takes a list of pairs and projects these pairs to their first components.

$$\begin{aligned}
\mathsf{maxlist}(xs, \mathsf{nil}) &\to \mathsf{nil} \\
\mathsf{maxlist}(\mathsf{nil}, \mathsf{cons}(y, ys)) &\to \mathsf{nil} \\
\mathsf{maxlist}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{cons}(x, \mathsf{maxlist}(xs, ys)) \qquad [\![x \geq y]\!] \\
\mathsf{maxlist}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{cons}(y, \mathsf{maxlist}(xs, ys)) \qquad [\![y > x]\!] \\
\mathsf{zip}(xs, \mathsf{nil}) &\to \mathsf{pnil} \\
\mathsf{zip}(\mathsf{nil}, \mathsf{cons}(y, ys)) &\to \mathsf{pnil} \\
\mathsf{zip}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{pcons}(\mathsf{pair}(x, y), \mathsf{zip}(xs, ys)) \\
\mathsf{fst}(\mathsf{pnil}) &\to \mathsf{nil} \\
\mathsf{fst}(\mathsf{pcons}(x, y), zs) &\to \mathsf{cons}(x, \mathsf{fst}(zs))
\end{aligned}$$

Consider the atomic conjecture $\mathsf{fst}(\mathsf{zip}(xs, xs)) \equiv \mathsf{maxlist}(xs, xs)$. The following derivation is a proof of this conjecture using the inference system $\mathcal{I}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle\ \{\mathsf{fst}(\mathsf{zip}(xs, xs)) \equiv \mathsf{maxlist}(xs, xs)\},\ \emptyset\ \rangle}{\langle\ \{\ \mathsf{fst}(\mathsf{pnil}) \equiv \mathsf{maxlist}(\mathsf{nil}, \mathsf{nil}),\ \mathsf{fst}(\mathsf{pcons}(\mathsf{pair}(x, x), \mathsf{zip}(xs, xs)) \equiv \mathsf{maxlist}(\mathsf{cons}(x, xs), \mathsf{cons}(x, xs))\ \},\ \{\ \mathsf{fst}(\mathsf{zip}(xs, xs)) \to \mathsf{maxlist}(xs, xs)\ \}\ \rangle}\ \mathsf{Expand}}{\langle\ \{\mathsf{nil} \equiv \mathsf{nil},\ \mathsf{cons}(x, \mathsf{fst}(\mathsf{zip}(xs, xs))) \equiv \mathsf{cons}(x, \mathsf{maxlist}(xs, xs))\},\ \{\mathsf{fst}(\mathsf{zip}(xs, xs)) \to \mathsf{maxlist}(xs, xs)\}\ \rangle}\ \mathsf{Simplify}^*}{\langle\ \{\mathsf{cons}(x, \mathsf{fst}(\mathsf{zip}(xs, xs))) \equiv \mathsf{cons}(x, \mathsf{maxlist}(xs, xs))\},\ \{\mathsf{fst}(\mathsf{zip}(xs, xs)) \to \mathsf{maxlist}(xs, xs)\}\ \rangle}\ \mathsf{Delete}}{\langle\ \{\mathsf{cons}(x, \mathsf{maxlist}(xs, xs)) \equiv \mathsf{cons}(x, \mathsf{maxlist}(xs, xs))\},\ \{\mathsf{fst}(\mathsf{zip}(xs, xs)) \to \mathsf{maxlist}(xs, xs)\}\ \rangle}\ \mathsf{Simplify}}{\langle\ \emptyset,\ \{\mathsf{fst}(\mathsf{zip}(xs, xs)) \to \mathsf{maxlist}(xs, xs)\}\ \rangle}\ \mathsf{Delete}$$

For the application of the inference rule Expand, notice that the second zip-rule does not produce any new atomic conjectures since the left-hand side of the second rule is not unifiable with $\mathsf{zip}(xs, xs)$. In $\mathsf{Simplify}^*$, only the rules from $\mathcal{R}$ are used for simplification. In the single Simplify step, the inductive hypothesis

[4] Strictly speaking, there is one predicate symbol $\simeq_s$ for each sort s. To simplify notation, these predicate symbols have been identified. Also, the predicate symbols $\geq$ and $>$ take two arguments of sort `int`.

from H is used. As stated in Theorem 28 below, this derivation shows that the atomic conjecture $\mathsf{fst}(\mathsf{zip}(xs, xs)) \equiv \mathsf{maxlist}(xs, xs)$ is an inductive theorem since the final state of the derivation has an empty set of atomic conjectures. △

The notation $\langle E, H\rangle \vdash_{\mathcal{I}} \langle E', H'\rangle$ is used to denote that the tuple $\langle E', H'\rangle$ has been obtained from $\langle E, H\rangle$ by one of the inference rules in Figure 1 and $\vdash^*_{\mathcal{I}}$ denotes the reflexive-transitive closure of $\vdash_{\mathcal{I}}$.

Next, several properties of the inference system $\mathcal{I}$ are shown. First, application of any inference rule leaves the convertibility relation of $\mathcal{R}$, E, H, and $\mathcal{E}_{\mathsf{LIA}}$ on ground terms unchanged. In particular, if $\langle E, \emptyset\rangle \vdash^*_{\mathcal{I}} \langle \emptyset, H\rangle$, then $\leftrightarrow^*_{\mathcal{R}\cup E\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} = \leftrightarrow^*_{\mathcal{R}\cup H\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ on ground terms.

Lemma 25. *If $\langle E_n, H_n\rangle \vdash_{\mathcal{I}} \langle E_{n+1}, H_{n+1}\rangle$ using an inference rule other than $\mathsf{Theory}_\perp$, then $\leftrightarrow^*_{\mathcal{R}\cup E_n\cup H_n\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} = \leftrightarrow^*_{\mathcal{R}\cup E_{n+1}\cup H_{n+1}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ on ground terms.*

The soundness proof of the inference system $\mathcal{I}$ is based on the following lemmas. The first lemma shows that if $\langle E, \emptyset\rangle \vdash^*_{\mathcal{I}} \langle \emptyset, H\rangle$, then each application of an atomic conjecture from E can be simulated by a "valley proof" using $\mathcal{R}$ and H.

Lemma 26. *If $\langle E_n, H_n\rangle \vdash^*_{\mathcal{I}} \langle \emptyset, H\rangle$ using inference rules other than $\mathsf{Theory}_\perp$, then $\leftrightarrow_{E_n,\mathbb{Z}} \subseteq \rightarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}}$ on ground terms.*

Using this property, the following statement can be shown. It relates the final set of hypotheses H to the rules of the $\mathbb{Z}$-TRS $\mathcal{R}$.

Lemma 27. *If $\langle E, \emptyset\rangle \vdash^*_{\mathcal{I}} \langle \emptyset, H\rangle$ using inference rules other than $\mathsf{Theory}_\perp$, then $\rightarrow_{H,\mathbb{Z}} \subseteq \rightarrow_{\mathcal{R},\mathbb{Z}} \circ \rightarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}}$ on ground terms.*

With these lemmas at hand, soundness of the inductive proof method based on the inference system $\mathcal{I}$ can be shown.

Theorem 28. *If $\langle E, \emptyset\rangle \vdash^*_{\mathcal{I}} \langle \emptyset, H\rangle$, then all atomic conjectures in E are inductive theorems.*

Example 29. This example considers the definition of $\mathsf{divides}$ from Example 3 and adds the following number-theoretic functions:

$$
\begin{array}{ll}
\mathsf{div}(x,y) \to -\mathsf{div}(-x,y) & [\![x < 0 \wedge y \geq 0]\!] \\
\mathsf{div}(x,y) \to -\mathsf{div}(x,-y) & [\![x \geq 0 \wedge y < 0]\!] \\
\mathsf{div}(x,y) \to -\mathsf{div}(-x,-y) & [\![x < 0 \wedge y < 0]\!] \\
\mathsf{div}(x,y) \to 0 & [\![x \geq 0 \wedge y \simeq 0]\!] \\
\mathsf{div}(x,y) \to 0 & [\![x \geq 0 \wedge y > 0 \wedge y > x]\!] \\
\mathsf{div}(x,y) \to \mathsf{div}(x-y,y) + 1 & [\![x \geq 0 \wedge y > 0 \wedge x \geq y]\!] \\
\mathsf{gcd}(x,y) \to \mathsf{gcd}(-x,y) & [\![x < 0 \wedge y \geq 0]\!] \\
\mathsf{gcd}(x,y) \to \mathsf{gcd}(x,-y) & [\![x \geq 0 \wedge y < 0]\!] \\
\mathsf{gcd}(x,y) \to \mathsf{gcd}(-x,-y) & [\![x < 0 \wedge y < 0]\!] \\
\mathsf{gcd}(x,y) \to y & [\![x \simeq 0 \wedge y \geq 0]\!] \\
\mathsf{gcd}(x,y) \to x & [\![x > 0 \wedge y \simeq 0]\!] \\
\mathsf{gcd}(x,y) \to \mathsf{gcd}(x-y,y) & [\![x > 0 \wedge y > 0 \wedge x \geq y]\!] \\
\mathsf{gcd}(x,y) \to \mathsf{gcd}(x,y-x) & [\![x > 0 \wedge y > 0 \wedge y > x]\!]
\end{array}
$$

Then, the conjectures

$$\begin{gathered}
\mathsf{div}(x,x) \equiv 1 \ [\![x \not\simeq 0]\!] \\
\mathsf{div}(x,y) \equiv -1 \ [\![y \not\simeq 0 \wedge y \simeq -x]\!] \\
\mathsf{div}(x,y) \equiv x \ [\![y \simeq 1]\!] \\
\{\ \mathsf{div}(x,y) \equiv -x \ [\![y \simeq -1]\!],\ \mathsf{div}(x,y) \equiv x \ [\![y \simeq 1]\!]\ \} \\
\mathsf{divides}(x,x) \equiv \mathsf{true} \\
\mathsf{divides}(x,y) \equiv \mathsf{true} \ [\![x \simeq -y]\!] \\
\mathsf{divides}(x,y) \equiv \mathsf{true} \ [\![x \simeq 1]\!] \\
\{\ \mathsf{divides}(x,y) \equiv \mathsf{true} \ [\![x \simeq -1]\!],\ \mathsf{divides}(x,y) \equiv \mathsf{true} \ [\![x \simeq 1]\!]\ \} \\
\mathsf{gcd}(x,x) \equiv x \ [\![x \geq 0]\!] \\
\mathsf{gcd}(x,x) \equiv -x \ [\![x \leq 0]\!] \\
\mathsf{gcd}(x,y) \equiv 1 \ [\![y \simeq 1]\!] \\
\{\ \mathsf{gcd}(x,y) \equiv 1 \ [\![y \simeq -1]\!],\ \mathsf{gcd}(x,y) \equiv 1 \ [\![y \simeq 1]\!]\ \}
\end{gathered}$$

can be proved fully automatically using the inference system $\mathcal{I}$. For the conjecture $\mathsf{divides}(x,x) \equiv \mathsf{true}$, a successful derivation is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle\ \{\mathsf{divides}(x,x) \equiv \mathsf{true}\},\ \emptyset\ \rangle}{\langle\ \{\ \mathsf{divides}(-x,-x) \equiv \mathsf{true}\ [\![x<0]\!],\ \mathsf{true} \equiv \mathsf{true}\ [\![x \simeq 0]\!],\ \mathsf{divides}(x,x-x) \equiv \mathsf{true}\ [\![x>0]\!]\ \},\ \{\ \mathsf{divides}(x,x) \to \mathsf{true}\ \}\ \rangle}\,\mathsf{Expand}}{\langle\ \{\ \mathsf{true} \equiv \mathsf{true}\ [\![x<0]\!],\ \mathsf{true} \equiv \mathsf{true}\ [\![x \simeq 0]\!],\ \mathsf{divides}(x,x-x) \equiv \mathsf{true}\ [\![x>0]\!]\ \},\ \{\ \mathsf{divides}(x,x) \to \mathsf{true}\ \}\ \rangle}\,\mathsf{Simplify}}{\langle\ \{\ \mathsf{true} \equiv \mathsf{true}\ [\![x<0]\!],\ \mathsf{true} \equiv \mathsf{true}\ [\![x \simeq 0]\!],\ \mathsf{true} \equiv \mathsf{true}\ [\![x>0]\!]\ \},\ \{\ \mathsf{divides}(x,x) \to \mathsf{true}\ \}\ \rangle}\,\mathsf{Simplify}}{\langle\ \emptyset,\ \{\mathsf{divides}(x,x)) \to \mathsf{true}\}\ \rangle}\,\mathsf{Delete}^3$$

Here, the constraints in the conjectures have been slightly simplified in order to ease presentation). △

The inference system $\mathcal{I}$ cannot only be used in order to prove inductive theorems. For this, it needs to be shown that the inference rule $\mathsf{Theory}_\bot$, which allows to disprove atomic conjectures, is sound.

Theorem 30. *If $\langle E, \emptyset\rangle \vdash_{\mathcal{I}}^* \bot$, then at least one atomic conjecture in E is not an inductive theorem.*

Example 31. Consider the atomic conjecture $\mathsf{fst}(\mathsf{zip}(xs,ys)) \equiv xs$, where fst and zip are the functions from Example 24. For this conjecture, the following derivation can be obtained using the inference system $\mathcal{I}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle\ \{\mathsf{fst}(\mathsf{zip}(xs,ys)) \equiv xs\},\ \emptyset\ \rangle}{\langle\ \{\mathsf{fst}(\mathsf{pnil}) \equiv xs, \mathsf{fst}(\mathsf{pnil}) \equiv \mathsf{nil}, \mathsf{fst}(\mathsf{pcons}(\mathsf{pair}(x,y),\mathsf{zip}(xs,ys))) \equiv \mathsf{cons}(x,xs)\},\ \{\mathsf{fst}(\mathsf{zip}(xs,ys)) \to xs\}\ \rangle}\,\mathsf{Expand}}{\langle\ \{\mathsf{nil} \equiv xs,\ \mathsf{nil} \equiv \mathsf{nil},\ \mathsf{cons}(x,xs) \equiv \mathsf{cons}(x,xs)\},\ \{\mathsf{fst}(\mathsf{zip}(xs,ys)) \to xs\}\ \rangle}\,\mathsf{Simplify}^*}{\langle\ \{\mathsf{nil} \equiv xs,\ \mathsf{nil} \equiv \mathsf{nil},\ \mathsf{cons}(x,xs) \equiv \mathsf{cons}(x,xs)\},\ \{\mathsf{fst}(\mathsf{zip}(xs,ys)) \to xs\}\ \rangle}\,\mathsf{Delete}^2}{\langle\ \{\mathsf{nil} \equiv xs\},\ \{\mathsf{fst}(\mathsf{zip}(xs,ys)) \to xs\}\ \rangle}\,\mathsf{Theory}_{\perp}}{\perp}$$

By Theorem 30, $\mathsf{fst}(\mathsf{zip}(xs, ys)) \equiv xs$ is not an inductive theorem. Furthermore, the derivation can be used to obtain a concrete counterexample for the conjecture: $xs = \mathsf{cons}(n, ns)$ and $ys = \mathsf{nil}$. △

Example 32. For the functions from Example 29, the conjectures

$$\mathsf{div}(x, x) \equiv \mathsf{1}$$
$$\mathsf{gcd}(x, x) \equiv x$$

can be disproved fully automatically using the inference system $\mathcal{I}$ (the first one is falsified by $x = \mathsf{0}$, the second one by any negative x). △

5 Inductive Theorem Proving as a Decision Procedure

While the inference system $\mathcal{I}$ from Section 4 provides a completely mechanical way to prove or disprove inductive conjectures once a strategy for the application of the inference rules has been fixed, it does not provide a decision procedure for inductive validity since derivations of the system may diverge or fail. The reason for a possible divergence is the inference rule Expand which could be applied again and again.

The goal of this section is to derive conditions on $\mathbb{Z}$-TRSs and conjectures under which the inference system $\mathcal{I}$ can be used as a decision procedure, i.e., will always produce a proof or disproof of a conjecture if a suitable strategy on the use of the inference rules is employed. These conditions are based on properties of the rewrite rules in a $\mathbb{Z}$-TRS that can be pre-computed during parsing. Thus, checking whether a conjecture satisfies the conditions under which $\mathcal{I}$ provides a decision procedure is easily possible and requires much less time than attempting a proof or disproof. Previous work on identifying conditions under which inductive theorem proving provides a decision procedure is discussed in Section 1.

Much of the material presented in this section has appeared in preliminary form in [15]. The use of $\mathbb{Z}$-TRSs and constrained rewriting is a significant generalization, however, since [15] was based on ordinary rewriting and did not support the combination of integers with (free) constructors (but was restricted to *either* natural numbers *or* (free) constructors).

5.1 Simple Decidable Conjectures

For the purpose of decidable induction, a simple class of function definitions is considered. In its simplest form, functions may only make recursive calls to themselves. Furthermore, nesting of recursive calls is not permitted. This is captured

by the following definition (this is related to the definition of $\mathcal{T}$-based functions in [29, 18]).

Definition 33 (LIAC-Based Functions). *A function $g \in \mathcal{D}(\mathcal{R})$ is* LIAC-based *iff all right-hand sides of rules in $\mathcal{R}(g)$ have the form $C[g(r_1^*), \ldots, g(r_m^*)]$ for a context C over $\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$ such that $r_k^* \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ for all $1 \leq k \leq m$.*

In order to ensure that a non-linear hypothesis is applicable to all recursive calls of a LIAC-based functions after application of the Expand rule, it needs to be ensured that the corresponding arguments of the recursive calls are "equal". More precisely, this needs to be required only under the assumption that these arguments are equal in the left-hand side of the rule since Expand does otherwise not create any new atomic conjectures to which the hypothesis needs to be applied. Notice that this property depends only on the rules in $\mathcal{R}(g)$ and is independent of the conjecture.

Definition 34 ($\mathcal{I}mp\mathcal{E}q$). *Let g be* LIAC*-based. Then $\langle i, j \rangle \in \mathcal{I}mp\mathcal{E}q(g)$ iff $1 \leq i < j \leq \mathrm{arity}(g)$ such that the i^{th} and j^{th} argument of g have the same sort and, for all $g(l^*) \to C[g(r_1^*), \ldots, g(r_m^*)][\![\varphi]\!] \in \mathcal{R}(g)$,*

$$\varphi \wedge l_i \simeq l_j \Rightarrow \bigwedge_{k=1}^{m} r_{k,i} \simeq r_{k,j}$$

is LIAC*-valid and $r_{k,i}, r_{k,j}$ are $\mathbb{Z}$-free for all $1 \leq k \leq m$.*

Hence, if a term of the form $g(l^*)\sigma$ is simplified using the rule $g(l^*) \to C[g(r_1^*), \ldots, g(r_m^*)][\![\varphi]\!]$ and $\langle i, j \rangle \in \mathcal{I}mp\mathcal{E}q(g)$, then $r_{k,i}\sigma = r_{k,j}\sigma$ for all $1 \leq k \leq m$ whenever $l_i\sigma = l_j\sigma$. The set $\mathcal{I}mp\mathcal{E}q(g)$ can easily be computed from the rules defining g with the help of a decision procedure for LIAC.

Example 35. The following orthogonal $\mathbb{Z}$-TRS determines whether a list is pointwise bigger than another list of the same length:

$$\begin{aligned}
\mathsf{ptwise}(\mathsf{nil}, \mathsf{nil}) &\to \mathsf{true} \\
\mathsf{ptwise}(\mathsf{nil}, \mathsf{cons}(y, ys)) &\to \mathsf{false} \\
\mathsf{ptwise}(\mathsf{cons}(x, xs), \mathsf{nil}) &\to \mathsf{false} \\
\mathsf{ptwise}(\mathsf{cons}(xs, xs), \mathsf{cons}(y, ys)) &\to \mathsf{ptwise}(xs, ys) \quad [\![x \geq y]\!] \\
\mathsf{ptwise}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{false} \quad [\![y > x]\!]
\end{aligned}$$

Then $\langle 1, 2 \rangle \in \mathcal{I}mp\mathcal{E}q(\mathsf{ptwise})$. So see this, notice that the implications from Definition 34 are trivially true for the first, second, third, and fifth rules since these rules do not contain any recursive calls. For the fourth rule, the LIAC-validity of

$$x \geq y \wedge \mathsf{cons}(x, xs) \simeq \mathsf{cons}(y, ys) \Rightarrow xs \simeq ys$$

is easily shown. △

Notice that the definition of $\mathcal{I}mp\mathcal{E}q$ requires that the argument are equal in all recursive calls in all rules. Using rewriting with constraints, this requirement can be relaxed so that more function definitions satisfy it. For recursive calls that can already by simplified to a term not containing the defined symbol, the $\mathcal{I}mp\mathcal{E}q$ requirement does not need to be satisfied since the inductive hypothesis does not need to be applied to this recursive call.

Example 36. Consider the function $\mathsf{divides}$ from Example 3 again. Then, the $\mathcal{I}mp\mathcal{E}q$ requirement is satisfied for the first through sixth rules. For the final rule, the $\mathcal{I}mp\mathcal{E}q$ requirement is not satisfied since $x > 0 \wedge y > 0 \wedge y \geq x \wedge x \simeq y \Rightarrow x \simeq y - x$ is *not* LIAC-valid. However, the atomic conjecture $\mathsf{divides}(x, x - x) \equiv \mathsf{true}\ [\![x > 0 \wedge x > 0 \wedge x \geq x]\!]$ generated in a proof attempt of $\mathsf{divides}(x, x) \equiv \mathsf{true}$ simplifies using $\rightarrow_{\mathcal{R},\mathbb{Z}}$ to $\mathsf{true} \equiv \mathsf{true}\ [\![x > 0 \wedge x > 0 \wedge x \geq x]\!]$ using the rewrite rule $\mathsf{divides}(x, y) \rightarrow \mathsf{true}\ [\![x \geq 0 \wedge y \simeq 0]\!]$. Thus, the inductive hypothesis does not need to be applied for this obligation. △

In order to be as general as possible, we consider simplification using the inference rule $\mathsf{Case\text{-}Simplify}$ (recall that $\mathsf{Case\text{-}Simplify}$ subsumes $\mathsf{Simplify}$ for rewriting using $\mathcal{R}$). The effect of repeated applications of $\mathsf{Case\text{-}Simplify}$ is captured by the following definition.

Definition 37 (Simplification Trees). *Let s be a term and let φ be a* LIA*-constraint. A* simplification tree *for $s[\![\varphi]\!]$ is a non-empty tree whose nodes are labelled with terms and* LIA*-constraints and whose root is labelled with $s[\![\varphi]\!]$ such that for every internal node labelled with $t[\![\psi]\!]$, the node has one child for every $t'[\![\psi']\!] \in \mathrm{Case}_p(t, \psi)$ and this child is labelled with $t'[\![\psi']\!]$.*

Using simplification trees, the set $\mathcal{I}mp\mathcal{E}q$ can be relaxed as follows.

Definition 38 ($\mathcal{I}mp\mathcal{E}q'$). *Let g be* LIAC*-based. Then $\langle i, j\rangle \in \mathcal{I}mp\mathcal{E}q'(g)$ iff $1 \leq i < j \leq \mathrm{arity}(g)$ such that the i^{th} and j^{th} arguments of g have the same sort and, for all rules $g(l^*) \rightarrow C[g(r_1^*), \ldots, g(r_m^*)][\![\varphi]\!] \in \mathcal{R}(g)$ and all $1 \leq k \leq m$, either*

1. *$\varphi \wedge l_i \simeq l_j \Rightarrow r_{k,i} \simeq r_{k,j}$ is* LIAC*-valid and $r_{k,i}, r_{k,j}$ are $\mathbb{Z}$-free, or*
2. *the i^{th} and j^{th} arguments of g have sort* `int` *and there exists a simplification tree for $g(r_k^*)[\![\varphi \wedge l_i \simeq l_j]\!]$ such that all leaves in this tree have labels the form $t[\![\psi]\!]$ for a term $t \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ and a* LIA*-constraint ψ.*

Example 39. Continuing Example 36,

$$\begin{array}{c} \mathsf{divides}(x, y - x)\ [\![x > 0 \wedge y > 0 \wedge y \geq x \wedge x \simeq y]\!] \\ \Big| \\ \mathsf{true}\ [\![x > 0 \wedge y > 0 \wedge y \geq x \wedge x \simeq y \wedge x \geq 0 \wedge y - x \simeq 0]\!] \end{array}$$

is a (degenerate) simplification tree that satisfies the conditions from case *2* in Definition 38. Thus, $\langle 1, 2\rangle \in \mathcal{I}mp\mathcal{E}q'(\mathsf{divides})$. Similarly, $\langle 1, 2\rangle \in \mathcal{I}mp\mathcal{E}q'(\mathsf{gcd})$ and $\langle 1, 2\rangle \in \mathcal{I}mp\mathcal{E}q'(\mathsf{div})$. △

Notice that $\mathcal{I}mp\mathcal{E}q'$ strictly subsumes $\mathcal{I}mp\mathcal{E}q$ and remains easily computable. The first version of decidable conjectures is now given as follows. Notice that only a simple form of basic terms is allowed, but that non-linearity is possible.

Definition 40 (Simple Conjectures). *A* simple conjecture *is an atomic conjecture of the form $g(x^*) \equiv t$ such that the following conditions are satisfied:*

1. *$\mathcal{R} \cup \{g(x^*) \to t\}$ is terminating.*
2. *The function g is* LIAC*-based.*
3. *x^* consists of variables and $t \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$.*
4. *Whenever $x_i = x_j$ for $i < j$, then $\langle i, j\rangle \in \mathcal{I}mp\mathcal{E}q'(g)$.*

Example 41. For the $\mathbb{Z}$-TRS from Example 35, the conjecture $\mathsf{ptwise}(xs, xs) \equiv \mathsf{true}$ is simple. For the $\mathbb{Z}$-TRS from Example 3, the conjecture $\mathsf{divides}(x, x) \equiv \mathsf{true}$ from Example 29 is simple. For the $\mathbb{Z}$-TRS from Example 29, the conjectures $\mathsf{div}(x, x) \equiv 1$ and $\mathsf{gcd}(x, x) \equiv x$ from Example 32 are simple. △

Theorem 42. *Using the strategy* $\mathsf{Expand} \cdot \mathsf{Case\text{-}Simplify}^* \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_\top \cup \mathsf{Theory}_\bot)^*$*, where* Simplify *uses only hypotheses from H, it is decidable whether a simple conjecture is an inductive theorem.*

The concept of LIAC-based functions is quite restrictive since a LIAC-based function may only make recursive calls to itself and not to any other function. The next definition generalizes this idea by considering a *set* of function symbols that may make recursive calls to each other (this is essentially the definition of jointly $\mathcal{T}$-based functions in [15]). Notice that nested recursive calls are not allowed, though.

Definition 43 (LIAC-Based Functions–Version 2). *A set of functions $\mathcal{G} = \{g_1, \ldots, g_n\} \subseteq \mathcal{D}(\mathcal{R})$ is* LIAC*-based iff all right-hand sides of rules in $\mathcal{R}(\mathcal{G})$ have the form $C[g_{k_1}(r_1^*), \ldots, g_{k_m}(r_m^*)]$ for some context C over $\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$ such that $r_i^* \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ and $g_{k_i} \in \mathcal{G}$ for all $1 \le i \le m$.*

Example 44. This example computes the pointwise average of two lists (stopping as soon as either list is empty) and uses the auxiliary function avg:

$$\begin{aligned}
\mathsf{avg}(x, y) &\to \mathsf{avg}(y, x) && [\![x > y]\!]\\
\mathsf{avg}(x, y) &\to x && [\![y \ge x \wedge y - x \le 1]\!]\\
\mathsf{avg}(x, y) &\to \mathsf{avg}(x + 1, y - 1) && [\![y \ge x \wedge y - x > 1]\!]\\
\mathsf{avglist}(xs, \mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{avglist}(\mathsf{nil}, \mathsf{cons}(y, ys)) &\to \mathsf{nil}\\
\mathsf{avglist}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{cons}(\mathsf{avg}(x, y), \mathsf{avglist}(xs, ys))
\end{aligned}$$

Since avglist makes a recursive call to avg, it is not LIAC-based. However, the set $\{\mathsf{avg}, \mathsf{avglist}\}$ is LIAC-based. △

In order to ensure that non-linear hypotheses are still applicable, the definition of $\mathcal{I}mp\mathcal{E}q$ and $\mathcal{I}mp\mathcal{E}q'$ needs to be adapted as well. For this, the idea is to collect conditions on all members of a LIAC-based set of functions under which

recursive calls to one of these functions g have equal arguments in positions i and j or can be rewritten to terms from $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$. These conditions are of the form $\langle g', i', j' \rangle$, meaning that equality of the arguments in positions i' and j' of the function g' ensures the desired property.

Definition 45 ($\mathcal{I}mp\mathcal{E}q$ and $\mathcal{I}mp\mathcal{E}q'$–Version 2). *Let $\mathcal{G} = \{g_1, \ldots, g_n\}$ be a* LIAC*-based set of functions. Then $\langle g, i, j, \Gamma \rangle \in \mathcal{I}mp\mathcal{E}q(\mathcal{G})$ for $g \in \mathcal{G}$ iff $1 \leq i < j \leq \mathrm{arity}(g)$ such that the i^{th} and j^{th} argument of g have the same sort and $\Gamma = \{\langle g_{k_1}, i_1, j_1 \rangle, \ldots, \langle g_{k_m}, i_m, j_m \rangle\}$ such that for all $1 \leq \kappa \leq m$, $1 \leq i_\kappa < j_\kappa \leq \mathrm{arity}(g_{k_\kappa})$ where the i_κ^{th} and j_κ^{th} argument of g_{k_κ} have the same sort, all rules $g_k(l^*) \to C[g_{b_1}(r_1^*), \ldots, g_{b_\omega}(r_\omega^*)][\![\varphi]\!] \in \mathcal{R}(\mathcal{G})$ satisfy that*

$$\varphi \wedge \bigwedge_{\langle g_k, i', j' \rangle \in \Gamma} l_{i'} \simeq l_{j'} \Rightarrow \bigwedge_{g_{b_\iota} = g} r_{\iota,i} \simeq r_{\iota,j}$$

is LIAC*-valid and all $r_{\iota,i}, r_{\iota,j}$ are $\mathbb{Z}$-free.*

Under the same conditions on i, j and Γ, let $\langle g, i, j, \Gamma \rangle \in \mathcal{I}mp\mathcal{E}q'(\mathcal{G})$ iff for all rules $g_k(l^*) \to C[g_{b_1}(r_1^*), \ldots, g_{b_\omega}(r_\omega^*)][\![\varphi]\!] \in \mathcal{R}(\mathcal{G})$ and all $1 \leq \iota \leq \omega$ with $g_{b_\iota} = g$, either

1. *$\varphi \wedge \bigwedge_{\langle g_k, i', j' \rangle \in \Gamma} l_{i'} \simeq l_{j'} \Rightarrow r_{\iota,i} \simeq r_{\iota,j}$ is* LIAC*-valid and $r_{\iota,i}, r_{\iota,j}$ are $\mathbb{Z}$-free, or*
2. *there exists a simplification tree for $g_{b_\iota}(r_\iota^*)[\![\varphi \wedge \theta]\!]$ where*

$$\theta = \bigwedge_{\langle g_k, i', j' \rangle \in \Gamma,\ \text{the } i'^{th} \text{ argument of } g \text{ has sort } \mathtt{int}} l_{i'} \simeq l_{j'}$$

such that all leaves in this tree have labels the form $t[\![\psi]\!]$ for a term $t \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ and a LIA*-constraint ψ.*

As before, the sets $\mathcal{I}mp\mathcal{E}q(\mathcal{G})$ and $\mathcal{I}mp\mathcal{E}q'(\mathcal{G})$ are still easily computable from the rules defining $\mathcal{G}$ with the help of a decision procedure for LIAC.

The definition of a simple conjecture immediately generalizes to LIAC-based sets $\mathcal{G}$ of functions. Now, an atomic conjecture for each member of the $\mathcal{G}$ is needed. Also, notice the use of $\mathcal{I}mp\mathcal{E}q'(\mathcal{G})$ to ensure applicability of the inductive hypotheses.

Definition 46 (Simple Conjectures–Version 2). *A* simple conjecture *is a set of atomic conjectures of the form $\{g_1(x_1^*) \equiv t_1, \ldots, g_n(x_n^*) \equiv t_n\}$ such that the following conditions are satisfied:*

1. *$\mathcal{R} \cup \{g_1(x_1^*) \to t_1, \ldots, g_n(x_n^*) \to t_n\}$ is terminating.*
2. *The set $\mathcal{G} = \{g_1, \ldots, g_n\}$ is* LIAC*-based.*
3. *x_i^* consists of variables and $t_i \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ for all $1 \leq i \leq n$.*
4. *Whenever $x_{k,i} = x_{k,j}$ for $i < j$, then there exists an $\langle g_k, i, j, \Gamma \rangle \in \mathcal{I}mp\mathcal{E}q'(\mathcal{G})$ such that $x_{k',i'} = x_{k',j'}$ for all $\langle g_{k'}, i', j' \rangle \in \Gamma$.*

Example 47. In Example 44, the set $\{\mathsf{avg}(x,x) \equiv x, \mathsf{avglist}(xs,xs) \equiv xs\}$ is a simple conjecture. To see this, notice that $\langle \mathsf{avglist}, 1, 2, \{\langle \mathsf{avglist}, 1, 2\rangle\}\rangle$ and $\langle \mathsf{avg}, 1, 2, \{\langle \mathsf{avg}, 1, 2\rangle, \langle \mathsf{avglist}, 1, 2\rangle\}\rangle$ are in $\mathcal{I}mp\mathcal{E}q(\{\mathsf{avg}, \mathsf{avglist}\})$ since

$$\mathsf{cons}(x, xs) \simeq \mathsf{cons}(y, ys) \Rightarrow xs \simeq ys$$
$$\mathsf{cons}(x, xs) \simeq \mathsf{cons}(y, ys) \Rightarrow x \simeq y$$

are LIAC-valid. △

Theorem 48. *Using the strategy* $\mathsf{Expand}^* \cdot \mathsf{Case\text{-}Simplify}^* \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_\top \cup \mathsf{Theory}_\bot)^*$*, where* Expand *is applied once to each atomic conjecture of the set and* Simplify *uses only hypotheses from H, it is decidable whether a simple conjecture is an inductive theorem.*

5.2 Simple Decidable Conjectures with Nesting

One restriction of the simple decidable conjectures from Section 5.1 is that nesting of defined function symbols is not permitted. This restriction was imposed in order to ensure that the inductive hypotheses are always applicable (if the $\mathcal{I}mp\mathcal{E}q'$-requirement is satisfied), resulting in an atomic conjecture whose validity can be decided using $\mathsf{Theory}_\top$ or $\mathsf{Theory}_\bot$.

For atomic conjectures with nested defined function symbols, this is not always the case since Expand might introduce a context from the right-hand sides of rules around the recursive calls. This context needs to be removed before the inductive hypotheses can be applied. This observation leads to the concept of *compatibility*, meaning that the $\mathbb{Z}$-TRS can handle the contexts introduced in right-hand sides of rules. The presentation in this section is influenced by the presentation in [18], which presents similar results for ordinary TRSs.

First, the following definition abstracts the defining property of LIAC-based functions and sets of LIAC-based functions.

Definition 49 (LIAC-Good Rewrite Rules and Functions). *A constrained rewrite rule $l \to r[\![\varphi]\!]$ is* LIAC-good *iff $r = C[g_1(r_1^*), \ldots, g_n(r_n^*)]$ where C is a context over $\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$ and $r_1^*, \ldots, r_n^* \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ for $g_1, \ldots, g_n \in \mathcal{D}(\mathcal{R})$. A function f is* LIAC-good *if all rules in $\mathcal{R}(f)$ are* LIAC-*good.*

In contrast to [18], the notion of compatibility is more complex and powerful in this paper since it is based on the inference rule Case-Simplify, i.e., it uses simplification trees. This makes the notion of compatibility more general, see Example 51 below.

Definition 50 (Compatibility). *Let g be* LIAC-*based, let $1 \leq j \leq \mathrm{arity}(g)$, and let $\mathcal{Q}$ be a set of* LIAC-*good rewrite rules. Then g* is compatible with $\mathcal{Q}$ on argument j *iff for all $f(l^*) \to C[g_1(r_1^*), \ldots, g_n(r_n^*)][\![\varphi]\!] \in \mathcal{Q}$ such that the j^{th} argument of g has the same sort as f, there exists a simplification tree*

for $g(x_1, \ldots, x_{j-1}, C[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m)[\![\varphi]\!]$ *such that all leaves in this tree have labels of the form*

$$D[g(x_1, \ldots, x_{j-1}, z_{i_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{i_k}, x_{j+1}, \ldots, x_m)][\![\varphi \wedge \psi]\!]$$

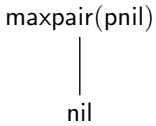
for a LIA*-constraint* ψ*, a context* D *over* $\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$*, and* $i_1, \ldots, i_k \in \{1, \ldots, n\}$ *such that* $z_i \notin \mathcal{V}(D)$ *for all* $1 \leq i \leq n$*.*

If $\mathcal{Q} = \mathcal{R}(f)$ for a function symbol f, then g is said to be compatible with f on argument j, and similarly for sets of function symbols.

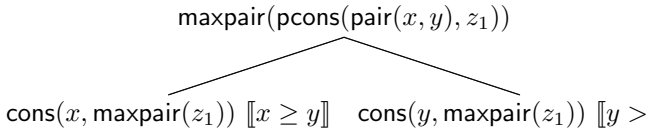
Example 51. Consider the $\mathbb{Z}$-TRS consisting of the zip-rules from Example 24 and add the following rules defining maxpair:

$$\begin{aligned} \mathsf{maxpair}(\mathsf{pnil}) &\to \mathsf{nil} \\ \mathsf{maxpair}(\mathsf{cons}(\mathsf{pair}(x, y), zs)) &\to \mathsf{cons}(x, \mathsf{maxpair}(zs))\ [\![x \geq y]\!] \\ \mathsf{maxpair}(\mathsf{cons}(\mathsf{pair}(x, y), zs)) &\to \mathsf{cons}(y, \mathsf{maxpair}(zs))\ [\![y > x]\!] \end{aligned}$$

Then maxpair is compatible with zip on argument 1. For the first two zip-rules, C is pnil (a context without holes), and the following is a (degenerate) simplification tree for maxpair(pnil):



The leaf has the required form by letting $D = \mathsf{nil}$. For the third zip-rule, C is $\mathsf{pcons}(\mathsf{pair}(x, y), \Box)$ and



is a simplification tree for $\mathsf{maxpair}(\mathsf{pcons}(\mathsf{pair}(x, y), z_1))$. Both leaves have the required form by letting $D = \mathsf{cons}(x, \Box)$ or $D = \mathsf{cons}(y, \Box)$, respectively.

Notice that the use of simplification trees is essential in this example, i.e., maxpair is *not* compatible with zip on argument 1 if the definition of compatibility from [18] is used. △

While Definition 50 considers the rules in $\mathcal{Q}$ independently for each context C from a right-hand side, the property from the definition can be lifted to nested contexts C that are obtained from several rules' right-hand sides. These contexts can be obtained if several rules from $\mathcal{Q}$ are applied after another.

Definition 52 (Repeated $\mathcal{Q}$-Contexts). *Let* $\mathcal{Q}$ *be a set of* LIAC*-good rewrite rules. For a context* C *and a* LIA*-constraint* φ*, the constrained context* $C[\![\varphi]\!]$ *is a* $\mathcal{Q}$-context *iff there exists a rule* $f(l^*) \to C[g_1(r_1^*), \ldots, g_n(r_n^*)][\![\varphi]\!] \in \mathcal{Q}$*. A constrained context* $C[\![\varphi]\!]$ *is a* repeated $\mathcal{Q}$-context *iff* $C[\![\varphi]\!]$ *is a* $\mathcal{Q}$*-context or there exist repeated* $\mathcal{Q}$*-contexts* $D[\![\psi]\!], C_1[\![\varphi_1]\!], \ldots, C_m[\![\varphi_m]\!]$ *such that* $C = D[C_1, \ldots, C_m][\![\psi \wedge \varphi_1 \wedge \ldots \wedge \varphi_m]\!]$*.*

If $\mathcal{Q} = \mathcal{R}(f)$, (repeated) $\mathcal{Q}$-contexts are also called (repeated) f-contexts, and similarly for sets of function symbols. Next, it can be shown that the property from Definition 50 lifts from the $\mathcal{Q}$-contexts considered there to repeated $\mathcal{Q}$-contexts.

Lemma 53. *Let g be compatible with $\mathcal{Q}$ on argument j. Then, for every repeated $\mathcal{Q}$-context $C_{\mathcal{Q}}[\![\varphi]\!]$, there exists a simplification tree for the constrained term $g(x_1, \ldots, x_{j-1}, C_{\mathcal{Q}}[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m)[\![\varphi]\!]$ such that all leaves in this tree have labels of the form*

$$C_g[g(x_1, \ldots, x_{j-1}, z_{i_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{i_k}, x_{j+1}, \ldots, x_m)][\![\varphi \wedge \psi]\!]$$

for a repeated g-context $C_g[\![\psi]\!]$ and $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that $z_i \notin \mathcal{V}(C_g)$ for all $1 \leq i \leq n$.

The concept of compatibility can be extended to arbitrarily deep nestings of functions, resulting in *compatibility sequences*.

Definition 54 (Compatibility Sequences). *Let $f_1, \ldots, f_{d-1}$ be* LIAC*-based and let f_d be* LIAC*-good for some $d \geq 1$. The sequence $\langle f_1, \ldots, f_d \rangle$ is a* compatibility sequence on arguments $\langle j_1, \ldots, j_{d-1} \rangle$ *iff f_i is compatible with f_{i+1} on argument j_i for all $1 \leq i \leq d-1$.*

A term s has *this compatibility sequence iff*

$$s = f_1(p_1^*, f_2(p_2^*, \ldots f_{d-1}(p_{d-1}^*, f_d(x^*), q_{d-1}^*) \ldots, q_2^*), q_1^*)$$

such that the variables in x^* do not occur elsewhere in s, the p_i^* and q_i^* are from $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$, and $f_i(p_i^*, f_{i+1}(\ldots), q_i^*)|_{j_i} = f_{i+1}(\ldots)$ for all $1 \leq i \leq d-1$.

If s is as in this definition, then $s\langle t \rangle$ denotes the term obtained from s by replacing the term $f_d(x^*)$ by the term t.

Lemma 55. *Let s be a term with the compatibility sequence $\langle f_1, \ldots, f_d \rangle$ on arguments $\langle j_1, \ldots, j_{d-1} \rangle$. Then, for every rule $f_d(l^*) \to C[g_1(r_1^*), \ldots, g_n(r_n^*)][\![\varphi]\!]$, there exists a simplification tree for $s\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)] \rangle [\![\varphi]\!]$ such that all leaves in this tree have labels of the form*

$$D[s\langle g_{i_1}(r_{i_1}^*) \rangle, \ldots, s\langle g_{i_k}(r_{i_k}^*) \rangle][\![\varphi \wedge \psi]\!]$$

for a LIA*-constraint ψ, a context D over $\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$, and some $i_1, \ldots, i_k \in \{1, \ldots, n\}$.*

Now *simple nested conjectures* generalize the simple conjectures from Section 5.1 by allowing nested defined functions on the left-hand side, provided the left-hand side has a compatibility sequence.

Definition 56 (Simple Nested Conjectures). *A* simple nested conjecture *is an atomic conjecture of the form $D[f(x^*)] \equiv t$ such that the following conditions are satisfied:*

1. $\mathcal{R} \cup \{D[f(x^*)] \to t\}$ *is terminating.*
2. *The term* $D[f(x^*)]$ *has a compatibility sequence and* f *is* LIAC*-based.*
3. x^* *consists of variables and* $t \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$.
4. *Whenever* $x_i = x_j$ *for* $i < j$, *then* $\langle i, j\rangle \in \mathit{Imp}\mathcal{E}q(f)$.

Notice that $\mathit{Imp}\mathcal{E}q(f)$ has to be used instead of $\mathit{Imp}\mathcal{E}q'(f)$ since, if the recursive calls to f can be simplified to terms from $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$, the remaining context D still prevents an application of $\mathsf{Theory}_\top$ or $\mathsf{Theory}_\bot$.

Example 57. Continuing Example 51, the term $\mathsf{maxpair}(\mathsf{zip}(xs, xs))$ has the compatibility sequence $\langle \mathsf{maxpair}, \mathsf{zip}\rangle$ on arguments $\langle 1\rangle$. Also, $\langle 1, 2\rangle \in \mathit{Imp}\mathcal{E}q(\mathsf{zip})$. Thus, $\mathsf{maxpair}(\mathsf{zip}(xs, xs)) \equiv xs$ is a simple nested conjecture. △

Theorem 58. *Using the strategy* $\mathsf{Expand} \cdot \mathsf{Case\text{-}Simplify}^* \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_\top \cup \mathsf{Theory}_\bot)^*$, *where* Simplify *uses only hypotheses from* H, *it is decidable whether a simple nested conjecture is an inductive theorem.*

Of course, the concept of simple nested conjectures can be extended from LIAC-based functions to LIAC-based sets of functions, similarly to how this was done for simple conjectures in Section 5.1. First, notice that the definition of compatibility can already be applied to LIAC-based sets of functions.

Example 59. Take the function fst defined in Example 24 and add the following rules defining stitch:

$$\begin{aligned}
\mathsf{stitch}(x, \mathsf{nil}) &\to \mathsf{pnil} \\
\mathsf{stitch}(\mathsf{nil}, \mathsf{cons}(y, ys)) &\to \mathsf{pnil} \\
\mathsf{stitch}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{pcons}(\mathsf{pair}(x, y), \mathsf{stitch}'(xs, ys)) \ [\![x \geq y]\!] \\
\mathsf{stitch}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{pcons}(\mathsf{pair}(y, x), \mathsf{stitch}'(xs, ys)) \ [\![y > x]\!] \\
\mathsf{stitch}'(x, \mathsf{nil}) &\to \mathsf{pnil} \\
\mathsf{stitch}'(\mathsf{nil}, \mathsf{cons}(y, ys)) &\to \mathsf{pnil} \\
\mathsf{stitch}'(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{pcons}(\mathsf{pair}(x, y), \mathsf{stitch}(xs, ys)) \ [\![x < y]\!] \\
\mathsf{stitch}'(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{pcons}(\mathsf{pair}(y, x), \mathsf{stitch}(xs, ys)) \ [\![y \leq x]\!]
\end{aligned}$$

Then $\mathcal{G} = \{\mathsf{stitch}, \mathsf{stitch}'\}$ is LIAC-based and fst is compatible with $\mathcal{G}$ on argument 1, since for the third and fourth stitch-rule, the term $\mathsf{fst}(\mathsf{pcons}(\mathsf{pair}(x, y), z_1))$ rewrites to $\mathsf{cons}(x, \mathsf{fst}(z_1))$, and similarly for the third and fourth stitch'-rule. △

Now the definition of simple nested conjectures can be revised as well, similarly how this was done for simple conjectures in Section 5.1.

Definition 60 (Simple Nested Conjectures–Version 2). *A* simple nested conjecture *is a set of the form* $\{D[f_1(x_1^*)] \equiv t_1, \ldots, D[f_n(x_n^*)] \equiv t_n\}$ *such that the following conditions are satisfied:*

1. $\mathcal{R} \cup \{D[f_1(x_1^*)] \to t_1, \ldots, D[f_n(x_n^*)] \to t_n\}$ *is terminating.*
2. *All of* $D[f_1(x_1^*)], \ldots, D[f_n(x_n^*)]$ *have compatibility sequences and the set* $\mathcal{G} = \{f_1, \ldots, f_n\}$ *is* LIAC*-based.*
3. x_i^* *consists of variables and* $t_i \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ *for all* $1 \leq i \leq n$.

4. *Whenever* $x_{k,i} = x_{k,j}$ *for* $i < j$*, then there exists an* $\langle f_k, i, j, \Gamma\rangle \in \mathcal{I}mp\mathcal{E}q(\mathcal{G})$ *such that* $x_{k',i'} = x_{k',j'}$ *for all* $\langle f_{k'}, i', j'\rangle \in \Gamma$*.*

Example 61. In Example 59, the term $\mathsf{fst}(\mathsf{stitch}(xs, xs))$ has the compatibility sequence $\langle \mathsf{fst}, \mathsf{stitch}\rangle$ on arguments $\langle 1\rangle$, and the term $\mathsf{fst}(\mathsf{stitch}'(xs, xs))$ has the compatibility sequence $\langle \mathsf{fst}, \mathsf{stitch}'\rangle$ on arguments $\langle 1\rangle$. Since $\{\mathsf{stitch}, \mathsf{stitch}'\}$ is LIAC-based, $\{\mathsf{fst}(\mathsf{stitch}(xs, xs)) \equiv xs, \mathsf{fst}(\mathsf{stitch}'(xs, xs)) \equiv xs\}$ is a simple nested conjecture because $\mathcal{I}mp\mathcal{E}q(\{\mathsf{stitch}, \mathsf{stitch}'\})$ contains $\langle \mathsf{stitch}, 1, 2, \{\langle \mathsf{stitch}', 1, 2\rangle\}\rangle$ and $\langle \mathsf{stitch}', 1, 2, \{\langle \mathsf{stitch}, 1, 2\rangle\}\rangle$. △

Theorem 62. *Using the strategy* $\mathsf{Expand}^* \cdot \mathsf{Case\text{-}Simplify}^* \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_\top \cup \mathsf{Theory}_\bot)^*$*, where* Expand *is applied once to each atomic conjecture of the set and* Simplify *uses only hypotheses from* H*, it is decidable whether a simple nested conjecture is an inductive theorem.*

6 Implementation and Evaluation

The inductive proof method based on the inference system $\mathcal{I}$ has been implemented in the prototype Sail2, the successor of Sail [15]. The implementation of the inference rules is mostly straightforward. Functions for checking whether a conjecture is simple or simple nested have been implemented in Sail2 as well. In order to perform these checks as efficiently as possible, the following information is pre-computed while parsing the $\mathbb{Z}$-TRS:

1. The sets $\mathcal{I}mp\mathcal{E}q(\mathcal{G})$ and $\mathcal{I}mp\mathcal{E}q'(\mathcal{G})$ are computed for each LIAC-based set of functions $\mathcal{G}$. This requires calls to CVC3 in order to check for LIAC-satisfiability and LIAC-validity.
2. Information on the compatibility between function symbols is computed. This is done using rewriting with the $\mathbb{Z}$-TRS, and in order to ensure that this rewriting process is terminating it is first checked whether the $\mathbb{Z}$-TRS is terminating.

In order to check for termination of $\mathcal{R} \cup H$ as needed in the side condition of Expand, the implementation of the methods for proving termination of CERSs developed in [14] in the termination tool AProVE is used. Since that implementation is currently limited to $\mathbb{Z}$-rules $l \to r[\![\varphi]\!]$ with $\mathcal{V}(r) \cup \mathcal{V}(\varphi) \subseteq \mathcal{V}(l)$, Sail2 imposes the same requirement on $\mathbb{Z}$-TRSs and atomic conjectures.[5] For validity and satisfiability checking of LIA- and LIAC-constraints, the external tools Yices [13] (for LIA-constraints) and CVC3 [6] (for LIAC-constraints) are used.

The implementation has been tested on 57 examples. This collection contains conjectures which can be shown to be decidable using the results from Section 5 and conjectures where this is not the case. The time spent for checking whether a conjecture is decidable as well as the time needed for (dis-)proving it have

[5] The termination methods developed in [14] do not require this condition on the variables but the implementation in AProVE does. Extending the implementation in AProVE should be straightforward but would require some re-engineering.

been recorded. Recall that a proof attempt requires a call to AProVE in order to determine whether the $\mathbb{Z}$-TRS together with the oriented conjectures is terminating. Also recall that a proof attempt requires calls to external SMT-solvers. The following table contains average times, the detailed results can be found at `http://baldur.iti.kit.edu/~falke/sail2/`.

Checking Time	SMT Time	Termination Time within AProVE	Other Time	Total Time
0.019 msec	16.451 msec	102.934 msec	0.554 msec	119.957 msec

As is immediate by inspection, the most time-consuming part (over 90% of the total time) is the termination check using AProVE. In contrast, checking whether a conjecture is a member of the class of decidable conjectures is orders of magnitude faster than proving or disproving it. Most of the time for the proof attempt is spent within the external SMT solvers. While the time spent within the SMT solvers can probably be reduced by using the SMT solver via its API (since this would eliminates the overhead of calling an external tool), the remaining parts of the proof attempt still require much more time than checking whether a conjecture is a member of the class of conjectures whose inductive validity is decidable. A more promising direction for improving the performance of Sail2 is to shift the burden of proving termination from AProVE to a more optimized implementation of the termination methods developed in [14].

7 Conclusions

We have presented new results on the decidability of validity for a class of conjectures that requires inductive reasoning. An implementation in the prototype Sail2 has been successfully evaluated on a collection of examples. This evaluation confirms that checking whether the inductive validity of a conjecture is decidable is indeed much faster than attempting to prove or disprove it. This paper further extends the results reported in [15] where decidability of inductive validity using the paradigm of implicit induction has been investigated for the first time.

The new decidability results reported in this paper were obtained using $\mathbb{Z}$-TRSs, for which an inductive proof method based on the implicit induction paradigm coupled with a decision procedure for the theory LIAC is given. The development of this proof method is a contribution in itself, independent of the decidability results about inductive validity of conjectures. The inductive proof method does not only make it possible to prove inductive conjectures, but also to disprove false conjectures.

There are two independent but related directions for future work. We are interested in developing an inductive proof method for more general classes of CERSs as defined in [14]. In contrast to $\mathbb{Z}$-TRSs, these CERSs also support (non-free) collection data structures such as sets or multisets and thus provide an even more expressive kind of term rewrite systems. In addition, we are planning to identify classes of conjectures with constraints and of conjectures containing nested function symbols on both sides whose inductive validity can be decided.

This may require techniques similar to [17, 18] which automatically generate suitable generalization lemmas that are needed for deciding validity. We believe that a combination of these techniques will lead to a very powerful decision procedure for proving inductive properties of algorithms specified on collection data structures such as sets or multisets.

A Proofs

Proof of Theorem 13. Let $\mathcal{R}$ be an orthogonal $\mathbb{Z}$-TRS and let $\overline{\mathcal{R}} = \{l \to r \mid l \to r[\![\varphi]\!] \in \mathcal{R}\}$ be the ordinary TRS obtained from $\mathcal{R}$ by dropping the constraints.

First, it is shown that $\mathcal{R}$ is quasi-reductive iff $\overline{\mathcal{R}}$ is quasi-reductive. The direction from left to right is immediate since $\to_{\mathcal{R},\mathbb{Z}} \subseteq \to_{\overline{\mathcal{R}}}$. For the direction from right to left, let $f(t_1,\dots,t_n)$ be a ground term with $f \in \mathcal{D}(\mathcal{R})$ and constructor ground terms $t_1,\dots,t_n$. Since $\overline{\mathcal{R}}$ is quasi-reductive, there exists a rule $l \to r \in \overline{\mathcal{R}}$ such that $f(t_1,\dots,t_n) = l\sigma$. By Definition 11.4, there exists a rule $l \to r'[\![\varphi]\!] \in \mathcal{R}$ such that $f(t_1,\dots,t_n) = l\sigma$ and $\varphi\sigma$ is LIA-valid. Therefore, $f(t_1,\dots,t_n)$ is reducible by $\to_{\mathcal{R},\mathbb{Z}}$.

It thus suffices to determine whether $\overline{\mathcal{R}}$ is quasi-reductive. But this can easily be done, for instance using the narrowing-based method for left-linear constructor-based ordinary TRSs in [20] that furthermore computes a set of missing patterns, i.e., left-hand sides for rules that need to be added in order to make $\mathcal{R}$ quasi-reductive. □

Proof sketch of Theorem 14. Orthogonal $\mathbb{Z}$-TRSs satisfy the following property:

> Whenever $s \to_{\mathcal{R},\mathbb{Z}} t_1$ and $s \to_{\mathcal{R},\mathbb{Z}} t_2$ such that the reductions take place at the same position, then $t_1 = t_2$.

Thus, orthogonal $\mathbb{Z}$-TRSs satisfy the crucial property needed to show confluence of orthogonal ordinary TRSs and the proof used for this result in [5, Corollary 6.3.11] immediately applies for orthogonal $\mathbb{Z}$-TRSs as well since orthogonal $\mathbb{Z}$-TRSs are left-linear by Definition 11.1.

To show the above property, it can without loss of generality be assumed that the reductions $s \to_{\mathcal{R},\mathbb{Z}} t_1$ and $s \to_{\mathcal{R},\mathbb{Z}} t_2$ take place at the root position. Thus, there exist rules $l_1 \to r_1[\![\varphi_1]\!], l_2 \to r_2[\![\varphi_2]\!]$ and substitutions σ_1, σ_2 such that $s = l_1\sigma_1 = l_2\sigma_2$ and $\varphi_1\sigma_1, \varphi_2\sigma_2$ are LIA-valid. By Definition 11.2, $l_1 = l_2$ and therefore $\sigma_1 = \sigma_2$. Now, Definition 11.3 implies that the rules $l_1 \to r_1[\![\varphi_1]\!]$ and $l_2 \to r_2[\![\varphi_2]\!]$ are identical, i.e., $t_1 = r_1\sigma_1 = r_2\sigma_2 = t_2$. □

Proof of Lemma 20. Let $s \equiv t[\![\varphi]\!]$ be an atomic conjecture and let u be a basic term such that $s = C[u]$.

1. Since u is basic and σ is a constructor ground substitution, the term $u\sigma$ has the form $f(u_1,\dots,u_n)$ where $u_1,\dots,u_n$ are constructor ground terms. Thus, since $\mathcal{R}$ is quasi-reductive, there exists a rule $l \to r[\![\psi]\!] \in \mathcal{R}$ such that $u\sigma = l\widehat{\sigma}$ and $\psi\widehat{\sigma}$ is LIA-valid for some $\mathbb{Z}$-based substitution $\widehat{\sigma}$. Without loss of generality it can be assumed that $\mathcal{V}(s) \cap \mathcal{V}(l) = \emptyset$, and the substitution σ may thus be extended to obtain $u\sigma = l\sigma$ such that $\psi\sigma$ is LIA-valid. Since σ is an unifier of u and l, there exists a $\mathbb{Z}$-based substitution θ such that $\sigma = \iota\theta$ where $\iota = \mathrm{mgu}(u,l)$. Thus, $s\sigma = C[u]\sigma = C[u]\iota\theta = C\iota\theta[u\iota\theta] = C\iota\theta[l\iota\theta] \to_{\mathcal{R},\mathbb{Z}} C\iota\theta[r\iota\theta] \leftrightarrow_{\mathrm{Expd}_u(s,t,\varphi),\mathbb{Z}} t\iota\theta = t\sigma$. For this, notice that $\varphi\iota\theta =$

$\varphi\sigma$ and $\psi\iota\theta = \psi\sigma$ are LIA-valid and that rewriting is closed under application of $\mathbb{Z}$-based substitutions.

2. Let $v \leftrightarrow_{\mathrm{Expd}_u(s,t,\varphi),\mathbb{Z}} w$. Then $v = \widehat{C}[C[r]\sigma\widehat{\sigma}]$ and $w = \widehat{C}[t\sigma\widehat{\sigma}]$ (or $w = \widehat{C}[C[r]\sigma\widehat{\sigma}]$ and $v = \widehat{C}[t\sigma\widehat{\sigma}]$) for some context $\widehat{C}$ and some $\mathbb{Z}$-based substitution $\widehat{\sigma}$, such that $\sigma = \mathrm{mgu}(u, l)$, $s = C[u]$, $l \to r[\![\psi]\!] \in \mathcal{R}$, and the LIA-constraint $\varphi\sigma\widehat{\sigma} \wedge \psi\sigma\widehat{\sigma}$ is LIA-valid. Then $v = \widehat{C}[C[r]\sigma\widehat{\sigma}] \leftarrow_{\mathcal{R},\mathbb{Z}} \widehat{C}[C[l]\sigma\widehat{\sigma}] = \widehat{C}[C\sigma[l\sigma]\widehat{\sigma}] = \widehat{C}[C\sigma[u\sigma]\widehat{\sigma}] = \widehat{C}[C[u]\sigma\widehat{\sigma}] = \widehat{C}[s\sigma\widehat{\sigma}] \to_{\{s \equiv t[\![\varphi]\!]\},\mathbb{Z}} \widehat{C}[t\sigma\widehat{\sigma}] = w$ since $\varphi\sigma\widehat{\sigma}$ and $\psi\sigma\widehat{\sigma}$ are LIA-valid. □

Proof of Lemma 25. Perform a case distinction according to the inference rule that is applied in $\langle E_n, H_n\rangle \vdash_{\mathcal{I}} \langle E_{n+1}, H_{n+1}\rangle$.

If this inference rule is Expand, the inclusion "$\subseteq$" is obvious. For "$\supseteq$", it suffices to show that $v \leftrightarrow_{\mathrm{Expd}_u(s,t,\varphi),\mathbb{Z}} w$ implies $v \leftrightarrow^*_{\mathcal{R}\cup E_n\cup H_n\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$ for all ground terms v, w. But this follows from Lemma 20.2 since $s \mathrel{\dot{\equiv}} t[\![\varphi]\!] \in E_n$.

For Simplify, $E_n = E \uplus \{s \mathrel{\dot{\equiv}} t[\![\varphi]\!]\}$, $E_{n+1} = E \cup \{s' \mathrel{\dot{\equiv}} t[\![\varphi]\!]\}$, and $H_{n+1} = H_n$, where $s[\![\varphi]\!] \to_{\mathcal{R}\cup H,\mathbb{Z}} s'[\![\varphi]\!]$. First, consider the inclusion "$\subseteq$". For this, assume $v \leftrightarrow^*_{\mathcal{R}\cup E_n\cup H_n\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$ for ground terms v, w. If $v \leftrightarrow^*_{\mathcal{R}\cup E\cup H_n\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$, then $v \leftrightarrow^*_{\mathcal{R}\cup E_{n+1}\cup H_{n+1}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$ is immediate. Otherwise, $v = C[s\sigma]$, $w = C[t\sigma]$ (or $w = C[s\sigma]$, $v = C[t\sigma]$), and $\varphi\sigma$ is LIA-valid for a $\mathbb{Z}$-based ground substitution σ. Now $s[\![\varphi]\!] \to_{\mathcal{R}\cup H_n,\mathbb{Z}} s'[\![\varphi]\!]$ implies that $s = D[l\tau]$, $s' = D[r\tau]$, and $\varphi \Rightarrow \psi\tau$ is LIA-valid for some $l \to r[\![\psi]\!] \in \mathcal{R} \cup H_n = \mathcal{R} \cup H_{n+1}$ and some $\mathbb{Z}$-based substitution τ. Since $\varphi \Rightarrow \psi\tau$ and $\varphi\sigma$ are LIA-valid, $\psi\tau\sigma$ is LIA-valid as well. Therefore, $v = C[s\sigma] = C[D\sigma[l\tau\sigma]] \to_{\mathcal{R}\cup H_{n+1},\mathbb{Z}} C[D\sigma[r\tau\sigma]] = C[s'\sigma] \to_{E_{n+1},\mathbb{Z}} C[t\sigma] = w$. For the inclusion "$\supseteq$", it again suffices to consider the case where $v = C[s'\sigma]$, $w = C[t\sigma]$ (or $w = C[s'\sigma]$, $v = C[t\sigma]$)), and $\varphi\sigma$ is LIA-valid. Similar to above, $v = C[s'\sigma] = C[D\sigma[r\tau\sigma]] \leftarrow_{\mathcal{R}\cup H_n,\mathbb{Z}} C[D\sigma[l\tau\sigma]] = C[s\sigma] \to_{E_n,\mathbb{Z}} C[t\sigma] = w$.

For the inference rule Case-Simplify, $E_n = E \uplus \{s \mathrel{\dot{\equiv}} t[\![\varphi]\!]\}$, $E_{n+1} = E \cup \{s' \equiv t[\![\varphi']\!] \mid s'[\![\varphi']\!] \in \mathrm{Case}_p(s, \varphi)\}$, and $H_{n+1} = H_n$. First, consider the inclusion "$\subseteq$". For this, assume $v \leftrightarrow_{\mathcal{R}\cup E_n\cup H_n\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$ for ground terms v, w. If $v \leftrightarrow_{\mathcal{R}\cup E\cup H_n\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$, then $v \leftrightarrow_{\mathcal{R}\cup E_{n+1}\cup H_{n+1}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$ is immediate. Otherwise, $v = C[s\sigma]$, $w = C[t\sigma]$ (or $w = C[s\sigma]$, $v = C[t\sigma]$), and $\varphi\sigma$ is LIA-valid for a $\mathbb{Z}$-based ground substitution σ. By the definition of $\mathrm{Case}_p(s, \varphi)$, there exists a $s[r_i\sigma_i]_p \equiv t[\![\varphi \wedge \psi_i\sigma_i]\!] \in E_{n+1}$ such that $\varphi\sigma \wedge \psi_i\sigma_i\sigma$ is LIA-valid. Therefore, $v = C[s\sigma] = C[s[l_i\sigma_i]_p\sigma] \to_{\mathcal{R},\mathbb{Z}} C[s[r_i\sigma_i]_p\sigma] \to_{E_{n+1},\mathbb{Z}} C[t\sigma] = w$. For the inclusion "$\supseteq$", it again suffices to consider the case where $v = C[s[r_i\sigma_i]_p\sigma]$, $w = C[t\sigma]$ (or $w = C[s[r_i\sigma_i]_p\sigma]$, $v = C[t\sigma]$)), and $\varphi\sigma \wedge \psi_i\sigma_i\sigma$ is LIA-valid. Similar to above, $v = C[s[r_i\sigma_i]_p\sigma] \leftarrow_{\mathcal{R},\mathbb{Z}} C[s[l_i\sigma_i]_p\sigma] = C[s\sigma] \to_{E_n,\mathbb{Z}} C[t\sigma] = w$.

For Delete, the inclusion "$\supseteq$" is obvious since an atomic conjecture is removed from E_n. For "$\subseteq$", it suffices to notice that $v \leftrightarrow_{\{s \mathrel{\dot{\equiv}} t[\![\varphi]\!]\},\mathbb{Z}} w$ for ground terms v and w implies that φ is LIA-satisfiable and that $v \leftrightarrow_{\{s \mathrel{\dot{\equiv}} t[\![\varphi]\!]\},\mathbb{Z}} w$ for ground terms v and w implies $v \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$ if $s \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t$ since all substitutions used in $v \leftrightarrow_{\{s \mathrel{\dot{\equiv}} t[\![\varphi]\!]\},\mathbb{Z}} w$ and in $s \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t$ are $\mathbb{Z}$-based.

For $\mathsf{Theory}_\top$, the inclusion "$\supseteq$" is again obvious. For "$\subseteq$", let $v \leftrightarrow_{\{s \mathrel{\dot{\equiv}} t[\![\varphi]\!]\},\mathbb{Z}} w$ for ground terms v and w. Thus, there exists a $\mathbb{Z}$-based ground substitution σ such that $v = C[s\sigma]$, $w = C[t\sigma]$ (or $w = C[s\sigma]$, $v = C[t\sigma]$), and $\varphi\sigma$ is

LIA-valid. Since $\mathcal{R}$ is quasi-reductive and terminating, there exists a constructor ground substitution $\widehat{\sigma}$ such that $\sigma(x) \to^*_{\mathcal{R},\mathbb{Z}} \widehat{\sigma}(x)$ for all variables x. Thus, $v = C[s\sigma] \to^*_{\mathcal{R},\mathbb{Z}} C[s\widehat{\sigma}] \leftrightarrow_{\{s \doteq t[\![\varphi]\!]\},\mathbb{Z}} C[t\widehat{\sigma}] \leftarrow^*_{\mathcal{R},\mathbb{Z}} C[t\sigma] = w$ and it suffices to show that $C[s\widehat{\sigma}] \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} C[t\widehat{\sigma}]$. Since $\varphi \Rightarrow s \simeq t$ is LIAC-valid and $\varphi\widehat{\sigma}$ is LIA-valid (since $\varphi\sigma$ is LIA-valid and $\sigma(x) = \widehat{\sigma}(x)$ for all variables x with sort `int` because σ is $\mathbb{Z}$-based), $s\widehat{\sigma} \simeq t\widehat{\sigma}$ is LIAC-valid as well. But this implies $s\widehat{\sigma} \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t\widehat{\sigma}$ and thus $C[s\widehat{\sigma}] \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} C[t\widehat{\sigma}]$. □

Proof of Lemma 26. The proof is by induction on the length k of the derivation $\langle E_n, H_n\rangle \vdash^*_{\mathcal{I}} \langle\emptyset, H\rangle$. If $k = 0$, then $E_n = \emptyset$ and the claim is obvious. Otherwise, $\langle E_n, H_n\rangle \vdash_{\mathcal{I}} \langle E_{n+1}, H_{n+1}\rangle \vdash^*_{\mathcal{I}} \langle\emptyset, H\rangle$. Now $\leftrightarrow_{E_{n+1},\mathbb{Z}} \subseteq \to^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}}$ follows from the inductive hypothesis. Next, perform a case distinction according to the inference rule that is applied in $\langle E_n, H_n\rangle \vdash_{\mathcal{I}} \langle E_{n+1}, H_{n+1}\rangle$.

If the inference rule Expand was applied, then $E_n = E \uplus \{s \doteq t[\![\varphi]\!]\}$, $E_{n+1} = E \cup \mathrm{Expd}_u(s, t, \varphi)$, and $H_{n+1} = H_n$. Let $v \leftrightarrow_{E_n,\mathbb{Z}} w$ for ground terms v, w. If $v \leftrightarrow_{E,\mathbb{Z}} w$, then the claim is immediate from the inductive hypothesis. Otherwise, $v = C[s\sigma]$ and $w = C[t\sigma]$ (or $v = C[t\sigma]$ and $w = C[s\sigma]$) where σ is $\mathbb{Z}$-based and $\varphi\sigma$ is LIA-valid. By Lemma 20.1, $v \to_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow_{\mathrm{Expd}_u(s,t,\varphi),\mathbb{Z}} w$ and thus $v \to_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow_{E_{n+1},\mathbb{Z}} w$. Now the inductive hypothesis implies $v \to^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} w$.

For the inference rule Simplify, $E_n = E \uplus \{s \doteq t[\![\varphi]\!]\}$, $E_{n+1} = E \cup \{s' \doteq t[\![\varphi]\!]\}$, and $H_{n+1} = H_n$, where $s[\![\varphi]\!] \to_{\mathcal{R}\cup H_n,\mathbb{Z}} s'[\![\varphi]\!]$. Let $v \leftrightarrow_{E_n,\mathbb{Z}} w$ for ground terms v, w. If $v \leftrightarrow_{E,\mathbb{Z}} w$, then the claim is immediate from the inductive hypothesis. Otherwise, $v \leftrightarrow_{\{s \doteq t[\![\varphi]\!]\},\mathbb{Z}} w$, i.e., $v = C[s\sigma]$ and $w = C[t\sigma]$ (or $v = C[t\sigma]$ and $w = C[s\sigma]$) where σ is $\mathbb{Z}$-based and $\varphi\sigma$ is LIA-valid. Since $v' := C[s'\sigma] \leftrightarrow_{E_{n+1},\mathbb{Z}} w$, the inductive hypothesis implies $v' \to^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} w$. It now suffices to show that $v \to_{\mathcal{R}\cup H,\mathbb{Z}} v'$ since then $v \to_{\mathcal{R}\cup H,\mathbb{Z}} v' \to^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} w$. To see this, recall that $s[\![\varphi]\!] \to_{\mathcal{R}\cup H,\mathbb{Z}} s'[\![\varphi]\!]$ implies $s = D[l\tau]$, $s' = D[r\tau]$, and $\varphi \Rightarrow \psi\tau$ is LIA-valid for some $l \to r[\![\psi]\!] \in \mathcal{R} \cup H_n$ and some $\mathbb{Z}$-based substitution τ. As in the proof of Lemma 25, $v = C[s\sigma] = C[D\sigma[l\tau\sigma]] \to_{\mathcal{R}\cup H_n,\mathbb{Z}} C[D\sigma[r\tau\sigma]] = C[s'\sigma] = v'$ and the claim follows since $H_n \subseteq H$.

For the inference rule Case-Simplify, $E_n = E \uplus \{s \doteq t[\![\varphi]\!]\}$, $E_{n+1} = E \cup \{s' \equiv t[\![\varphi']\!] \mid s'[\![\varphi']\!] \in \mathrm{Case}_p(s, \varphi)\}$, and $H_{n+1} = H_n$. Let $v \leftrightarrow_{E_n,\mathbb{Z}} w$ for ground terms v, w. If $v \leftrightarrow_{E,\mathbb{Z}} w$, then the claim is immediate from the inductive hypothesis. Otherwise, $v \leftrightarrow_{\{s \doteq t[\![\varphi]\!]\},\mathbb{Z}} w$, i.e., $v = C[s\sigma]$ and $w = C[t\sigma]$ (or $v = C[t\sigma]$ and $w = C[s\sigma]$) where σ is $\mathbb{Z}$-based and $\varphi\sigma$ is LIA-valid. By the definition of $\mathrm{Case}_p(s, \varphi)$, there exists a $s[r_i\sigma_i]_p \equiv t[\![\varphi \wedge \psi_i\sigma_i]\!] \in E_{n+1}$ such that $\varphi\sigma \wedge \psi_i\sigma_i\sigma$ is LIA-valid. Since $v' := C[s[r_i\sigma_i]_p\sigma] \leftrightarrow_{E_{n+1},\mathbb{Z}} w$, the inductive hypothesis implies $v' \to^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} w$. It now suffices to show that $v \to_{\mathcal{R}\cup H,\mathbb{Z}} v'$ since then $v \to_{\mathcal{R}\cup H,\mathbb{Z}} v' \to^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} w$. As in the proof of Lemma 25, $v = C[s\sigma] = C[s[l_i\sigma_i]_p\sigma] \to_{\mathcal{R},\mathbb{Z}} C[s[r_i\sigma_i]_p\sigma] = v'$ and the claim of the lemma therefore follows.

If the inference rule Delete was applied, then $E_n = E \uplus \{s \doteq t[\![\varphi]\!]\}$ where $s \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t$ or φ is LIA-unsatisfiable, $E_{n+1} = E$, and $H_{n+1} = H_n$. If $v \leftrightarrow_{E,\mathbb{Z}} w$, then the claim follows from the inductive hypothesis. If φ is LIA-unsatisfiable,

then $s \doteq t[\![\varphi]\!]$ cannot contribute to $\leftrightarrow_{E_n,\mathbb{Z}}$. Otherwise, $v \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} w$ as in the proof of Lemma 25 and the claim is immediate.

If the inference rule $\mathsf{Theory}_\top$ was applied, then $E_n = E \uplus \{s \doteq t[\![\varphi]\!]\}$, $E_{n+1} = E$, and $H_{n+1} = H_n$, where $\varphi \Rightarrow s \simeq t$ is LIAC-valid. Again, if $v \leftrightarrow_{E,\mathbb{Z}} w$, then the claim follows from the inductive hypothesis. Otherwise, $v = C[s\sigma]$ and $w = C[t\sigma]$ (or $v = C[t\sigma]$ and $w = C[s\sigma]$) where $\varphi\sigma$ is LIA-valid. As in the proof of Lemma 25, $v \rightarrow^*_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R},\mathbb{Z}} w$ and the claim is immediate. □

Proof of Lemma 27. Let $s \to t[\![\varphi]\!] \in H$ and $v \rightarrow_{\{s \to t[\![\varphi]\!]\},\mathbb{Z}} w$ for ground terms v and w, i.e., $v = C[s\sigma]$ and $w = C[t\sigma]$ for a $\mathbb{Z}$-based ground substitution σ such that $\varphi\sigma$ is LIA-valid. Since $\mathcal{R}$ is quasi-reductive and terminating, there exists a constructor ground substitution $\widehat{\sigma}$ such that $\sigma(x) \rightarrow^*_{\mathcal{R},\mathbb{Z}} \widehat{\sigma}(x)$ for all variables x. Then, $C[s\sigma] \rightarrow^*_{\mathcal{R},\mathbb{Z}} C[s\widehat{\sigma}] \rightarrow_{\{s \to t[\![\varphi]\!]\},\mathbb{Z}} C[t\widehat{\sigma}] \leftarrow^*_{\mathcal{R},\mathbb{Z}} C[t\sigma]$ since $\varphi\widehat{\sigma}$ is LIA-valid because $\sigma(x) = \widehat{\sigma}(x)$ for all variables x of sort `int`. It thus suffices to show $C[s\widehat{\sigma}] \rightarrow^*_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R},\mathbb{Z}} C[t\widehat{\sigma}]$.

There exists an n such that $\langle E, \emptyset\rangle \vdash^*_{\mathcal{I}} \langle E_n, H_n\rangle \vdash_{\mathcal{I}} \langle E_{n+1}, H_{n+1}\rangle \vdash^*_{\mathcal{I}} \langle \emptyset, H\rangle$ where $H_{n+1} = H_n \cup \{s \to t[\![\varphi]\!]\}$, $E_n = E'_n \uplus \{s \equiv t[\![\varphi]\!]\}$, and $E_{n+1} = E'_n \cup \mathrm{Expd}_u(s,t,\varphi)$. Then, by Lemma 20.1, $s\widehat{\sigma} \rightarrow_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow_{\mathrm{Expd}_u(s,t,\varphi),\mathbb{Z}} t\widehat{\sigma}$ and thus $C[s\widehat{\sigma}] \rightarrow_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow_{E_{n+1},\mathbb{Z}} C[t\widehat{\sigma}] = w$. But then Lemma 26 gives the desired $C[s\widehat{\sigma}] \rightarrow_{\mathcal{R},\mathbb{Z}} \circ \rightarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} C[t\widehat{\sigma}]$. □

Proof of Theorem 28. By Lemma 25, $\leftrightarrow^*_{\mathcal{R}\cup E\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} = \leftrightarrow^*_{\mathcal{R}\cup H\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ on ground terms. Thus, it suffices to show that $\leftrightarrow^*_{\mathcal{R}\cup H\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} = \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ on ground terms. For this, the following principle is used:

> Assume that the following conditions are satisfied:
>
> 1. $\rightarrow_{\mathcal{R}\cup H,Z}$ is terminating on ground terms.
> 2. $\rightarrow_{H,\mathbb{Z}} \subseteq \rightarrow_{\mathcal{R},\mathbb{Z}} \circ \rightarrow^*_{\mathcal{R}\cup H,Z} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,Z}$ on ground terms.
>
> Then $\leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} = \leftrightarrow^*_{\mathcal{R}\cup H\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ on ground terms.

This principle is quite similar to an abstract principle of Koike and Toyama [32] as reported in [1, 2] but differs from that principle by incorporating $\mathcal{E}_{\mathsf{LIA}}$.

With this principle, the statement of the theorem can be shown. The first condition, i.e., that $\rightarrow_{\mathcal{R}\cup H,\mathbb{Z}}$ is terminating on ground terms, follows from the assumption on $\mathcal{R}$ and from the condition of the inference rule Expand. The second condition is the property from Lemma 27.

Thus, it remains to show correctness of the principle. The inclusion "$\subseteq$" is obvious. For "$\supseteq$", let T denote the set of all ground terms. It suffices to show that for any $x \in T$, $\forall y \in T.\ x \rightarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} y \Rightarrow x \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} y$ is true. This property is shown by Noetherian induction on $\rightarrow_{\mathcal{R}\cup H,\mathbb{Z}}$ which, by condition *1*, is well-founded on ground terms. Thus, let $x \rightarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} y$. If $x = y$, then $x \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} y$ is obvious. Otherwise, $x \rightarrow_{\mathcal{R}\cup H,\mathbb{Z}} z \rightarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} y$ and hence $z \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} y$ by the inductive hypothesis. If $x \rightarrow_{\mathcal{R},\mathbb{Z}} z$, then $x \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} y$ is immediate. Otherwise, $x \rightarrow_{H,\mathbb{Z}} z$. By condition *2*, there exists a c such that $x \rightarrow_{\mathcal{R},\mathbb{Z}} c \rightarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R}\cup H,\mathbb{Z}} z$. Applying the induction hypothesis to c

and z then yields $x \to_{\mathcal{R},\mathbb{Z}} c \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} z$, i.e., $x \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} z$. Together with $z \leftrightarrow_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} y$, this gives $x \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} y$ as desired. □

Proof of Theorem 30. Let $\langle E, \emptyset\rangle \vdash^*_{\mathcal{I}} \langle E_n, H_n\rangle \vdash_{\mathcal{I}} \bot$ where $E_n = E'_n \uplus \{s \doteq t[\![\varphi]\!]\}$ such that s, t do not contain symbols from $\mathcal{D}(\mathcal{R})$ and $\varphi \Rightarrow s \simeq t$ is not LIAC-valid. Therefore, there exists a constructor ground substitution σ such that $\varphi\sigma$ is LIA-valid and $s\sigma \simeq t\sigma$ is not LIAC-valid, which implies $s\sigma \not\leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t\sigma$. Since $s\sigma \leftrightarrow_{E_n,\mathbb{Z}} t\sigma$, Lemma 25 implies $s\sigma \leftrightarrow^*_{\mathcal{R}\cup E\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t\sigma$. If all atomic conjectures in E are inductive theorems, then this implies $s\sigma \leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t\sigma$. If $\to_{\mathcal{R},\mathbb{Z}}$ is Church-Rosser modulo $\leftrightarrow_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ (i.e., $\leftrightarrow_{\mathcal{R}\cup\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \subseteq \to^*_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R},\mathbb{Z}}$), then this implies $s\sigma \to^*_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \leftarrow^*_{\mathcal{R},\mathbb{Z}} t\sigma$. But since $s\sigma$ and $t\sigma$ are irreducible by $\to_{\mathcal{R},\mathbb{Z}}$ since s and t do not contain defined symbols and σ is a constructor ground substitution, this yields $s\sigma \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} t\sigma$, i.e., a contradiction.

Thus, it remains to be shown that $\to_{\mathcal{R},\mathbb{Z}}$ is Church-Rosser modulo $\leftrightarrow_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$. Since $\mathcal{R}$ is terminating and confluent (and thus confluent modulo $\leftrightarrow_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$), results from [23] imply that it suffices to show that $\to_{\mathcal{R},\mathbb{Z}}$ is strongly coherent modulo $\leftrightarrow_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$, i.e., that $\leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \to_{\mathcal{R},\mathbb{Z}} \subseteq \to_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$. For this, we show that $\leftrightarrow_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} \circ \to_{\mathcal{R},\mathbb{Z}} \subseteq \to_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ holds. Thus, let $s_1 \leftrightarrow_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} s_2 \to_{\mathcal{R},\mathbb{Z}} s_3$, i.e., there exist $l_1 \approx r_1 \in \mathcal{E}_{\mathsf{LIA}}$ (or $r_1 \approx l_1 \in \mathcal{E}_{\mathsf{LIA}}$), $l_2 \to r_2[\![\varphi]\!] \in \mathcal{R}$, positions $p_1 \in \mathcal{P}os(s_1)$ and $p_2 \in \mathcal{P}os(s_2)$, and $\mathbb{Z}$-based substitutions σ_1 and σ_2 such that $s_1|_{p_1} = l_1\sigma_1$, $s_2 = s_1[r_1\sigma_1]_{p_1}$, $s_2|_{p_2} = l_2\sigma_2$, $s_3 = s_2[r_2\sigma_2]_{p_2}$, and $\varphi\sigma_2$ is LIA-valid. Perform a case distinction on the relationship between p_1 and p_2.

If p_1 and p_2 are independent of each other, then $s_1 \to_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}}$ is immediate. Notice that p_1 cannot be above or equal to p_2 since σ_1 is $\mathbb{Z}$-based. Finally, assume that p_1 is strictly below p_2, i.e., $p_1 = p_2.q$ for some non-empty position q. Notice that the position q is "inside" σ_2 since l_1 is $\mathbb{Z}$-free and σ_1 and σ_2 are $\mathbb{Z}$-based. But then $s_1 \to_{\mathcal{R},\mathbb{Z}} \circ \leftrightarrow^*_{\mathcal{E}_{\mathsf{LIA}},\mathbb{Z}} s_2$ is immediate as well. □

Proof of Theorem 42. Let $g(x^*) \equiv t$ be a simple conjecture and consider a rule $g(l^*) \to C[g(r^*_1), \ldots, g(r^*_m)][\![\varphi]\!] \in \mathcal{R}(g)$. Provided $g(x^*)$ and $g(l^*)$ are unifiable and $\varphi\sigma$ is LIA-satisfiable for $\sigma = \mathrm{mgu}(g(x^*), g(l^*))$, application of Expand to $g(x^*) \equiv t$ produces (amongst others) $C\sigma[g(r^*_1)\sigma, \ldots, g(r^*_m)\sigma] \equiv t\sigma[\![\varphi\sigma]\!]$. After application of Expand, the set H of hypotheses consists of the oriented conjecture $g(x^*) \to t$.

Now, if $x_i = x_j$ for $i < j$, then $\langle i, j\rangle \in \mathcal{I}mp\mathcal{E}q'(g)$. Thus, since $x_i = x_j$ implies $l_i\sigma = l_j\sigma$, the definition of $\mathcal{I}mp\mathcal{E}q'$ yields, for all $1 \le k \le m$, that either $r_{k,i}\sigma = r_{k,j}\sigma$ or there exists a simplification tree for $g(r^*_k)[\![\varphi \wedge l_i \simeq l_j]\!]$ such that all leaves in this tree have labels of the form $t[\![\psi]\!]$ for a $t \in \mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ and a LIA-constraint ψ. Hence, $\mathsf{Case\text{-}Simplify}$ and/or $\mathsf{Simplify}$ using the hypothesis $g(x^*) \to t \in H$ can be applied to the conjecture $C\sigma[g(r^*_1)\sigma, \ldots, g(r^*_m)^*\sigma] \equiv t\sigma[\![\varphi\sigma]\!]$ to obtain $C\sigma[q_1, \ldots, q_m] \equiv t\sigma[\![\varphi\sigma]\!]$, where q_i is either a term in $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$ or $q_i = t\tau_i$ with $\tau_i = \{x^* \mapsto r^*_i\sigma\}$. Since both sides are from $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$, either $\mathsf{Theory}_\top$ or $\mathsf{Theory}_\bot$ can be applied. □

Proof of Theorem 48. Let $\{g_1(x^*_1) \equiv t_1, \ldots, g_n(x^*_n) \equiv t_n\}$ be a simple set of conjectures. Consider the atomic conjecture $g_k(x^*_k) \equiv t_k$ from this set and the

rule $g_k(l^*) \to C[g_{k_1}(r_1^*), \ldots, g_{k_m}(r_m^*)][\![\varphi]\!] \in \mathcal{R}(\mathcal{G})$. Provided $g_k(x_k^*)$ and $g_k(l^*)$ are unifiable and $\varphi\sigma$ is LIA-satisfiable for $\sigma = \mathrm{mgu}(g_k(x_k^*), g_k(l^*))$, application of Expand to $g_k(x_k^*) \equiv t_k$ produces (amongst others) $C\sigma[g_{k_1}(r_1^*)\sigma, \ldots, g_{k_m}(r_m^*)\sigma] \equiv t_k\sigma[\![\varphi\sigma]\!]$. After application of Expand to each atomic conjecture in the simple set of conjectures, the set H consists of the rules $g_1(x_1^*) \to t_1, \ldots, g_n(x_n^*) \to t_n$.

Now, if $x_{k_\kappa,i_\kappa} = x_{k_\kappa,j_\kappa}$ for any $1 \leq \kappa \leq m$ and $i_\kappa < j_\kappa$, there exists an $\langle g_{k_\kappa}, i_\kappa, j_\kappa, \Gamma\rangle \in \mathcal{I}mp\mathcal{E}q(\mathcal{G})$ such that $x_{k',i'} = x_{k',j'}$ for all $\langle g_{k'}, i', j'\rangle \in \Gamma$. In particular, all such restrictions for g_k are satisfied. As in the proof of Theorem 42, the definition of $\mathcal{I}mp\mathcal{E}q'$ implies that $\mathsf{Case\text{-}Simplify}$ and/or $\mathsf{Simplify}$ can be applied to $C\sigma[g_{k_1}(r_1^*)\sigma, \ldots, g_{k_m}(r_m^*)^*\sigma] \equiv t_k\sigma[\![\varphi\sigma]\!]$, resulting in a conjecture where both sides of are from $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$. Finally, either $\mathsf{Theory}_\top$ or $\mathsf{Theory}_\bot$ can be applied. □

Proof of Lemma 53. The statement is proved by induction on $C_\mathcal{Q}[\![\varphi]\!]$. If $C_\mathcal{Q}[\![\varphi]\!]$ is a $\mathcal{Q}$-context, then Definition 50 implies that there exists a simplification tree for $g(x_1, \ldots, x_{j-1}, C_\mathcal{Q}[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m)[\![\varphi]\!]$ such that all leaves of this tree have the form

$$D[g(x_1, \ldots, x_{j-1}, z_{i_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{i_k}, x_{j+1}, \ldots, x_m)][\![\varphi \wedge \psi]\!]$$

such that $z_i \notin \mathcal{V}(D)$ for all $1 \leq i \leq n$. Thus, it only remains to be shown that $D[\![\psi]\!]$ is a repeated g-context. But this easily follows since g is LIAC-based and $C_\mathcal{Q}$ only contains symbols from $\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$.

If $C_Q[\![\varphi]\!]$ is a repeated $\mathcal{Q}$-context of the form $C[C_1, \ldots, C_k][\![\psi \wedge \varphi_1 \wedge \ldots \wedge \varphi_k]\!]$ for repeated $\mathcal{Q}$-contexts $C[\![\psi]\!], C_1[\![\varphi_1]\!], \ldots, C_k[\![\varphi_k]\!]$, then the inductive hypothesis implies that there exists a simplification tree for the constrained term $g(x_1, \ldots, x_{j-1}, C_\mathcal{Q}[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m)[\![\varphi]\!]$, i.e., for the constrained term $g(x_1, \ldots, x_{j-1}, C[C_1, \ldots, C_k][z_1, \ldots, z_n], x_{j+1}, \ldots, x_m)[\![\psi \wedge \varphi_1 \wedge \ldots \wedge \varphi_k]\!]$, such that all leaves in this tree have the form

$$D[g(x_1, \ldots, x_{j-1}, u_1, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, u_d, x_{j+1}, \ldots, x_m)] \\ [\![\varphi \wedge \varphi' \wedge \psi \wedge \varphi_1 \wedge \ldots \wedge \varphi_k]\!]$$

where $D[\![\varphi']\!]$ is a repeated g-context and $z_i \notin \mathcal{V}(D)$ for all $1 \leq i \leq n$. Here, $u_l = C_{e_l}[z_1, \ldots, z_n]$ for all $1 \leq l \leq d$. Furthermore, the inductive hypothesis implies that there exist repeated g-context $D_1[\![\varphi_1']\!], \ldots, D_d[\![\varphi_d']\!]$ with $z_i \notin \mathcal{V}(D_l)$ for all $1 \leq i \leq n$ and $1 \leq l \leq d$ such that there exists a simplification tree for $g(x_1, \ldots, x_{j-1}, u_l, x_{j+1}, \ldots, x_m)[\![\varphi_l]\!]$ such that all leaves in this tree have the form

$$D_l[g(x_1, \ldots, x_{j-1}, z_{l_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{l_{k_l}}, x_{j+1}, \ldots, x_m)][\![\varphi_l \wedge \varphi_l']\!]$$

for all $1 \leq l \leq d$, where $l_1, \ldots, l_{k_l} \in \{1, \ldots, n\}$. Therefore, there exists a simplification tree for $g(x_1, \ldots, x_{j-1}, C_\mathcal{Q}[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m)[\![\varphi]\!]$ such that all leaves in this tree have the form

$$D[D_1, \ldots, D_d][g(x_1, \ldots, x_{j-1}, z_{1_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{d_{k_d}}, x_{j+1}, \ldots, x_m)] \\ [\![\varphi \wedge \varphi' \wedge \psi \wedge \varphi_1 \wedge \ldots \wedge \varphi_k \wedge \varphi_1' \wedge \varphi_d']\!]$$

where $C_g[\![\theta]\!] := D[D_1, \ldots, D_d][\![\varphi' \wedge \varphi'_1 \wedge \ldots \wedge \varphi'_d]\!]$ is a repeated g-context. Furthermore, $\mathcal{V}(C_g) = \mathcal{V}(D[D_1, \ldots, D_d]) = \mathcal{V}(D) \cup \mathcal{V}(D_1) \cup \ldots \cup \mathcal{V}(D_d)$ does not contain any z_i for $1 \leq i \leq n$. □

Proof of Lemma 55. Define a sequence of terms by $s_d = f_d(x_{d,1} \ldots, x_{d,m_d})$ and $s_i = f_i(x_{i,1}, \ldots, x_{i,j_i-1}, s_{i+1}, x_{i,j_i+1}, \ldots, x_{i,m_i})$ for all $1 \leq i \leq d-1$. The lemma is proved by showing the following statement for all $1 \leq i \leq d$:

(†) For the constrained term $s_i\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle[\![\varphi]\!]$, there exists a simplification tree such that all leaves in this tree have labels of the form $D[s_i\langle g_{j_1}(r_{j_1}^*)\rangle, \ldots, s_i\langle g_{j_l}(r_{j_l}^*)\rangle][\![\varphi \wedge \psi]\!]$ for some $j_1, \ldots, j_l \in \{1, \ldots, n\}$ and a repeated f_i-context $D[\![\psi]\!]$ with $\mathcal{V}(D) \subseteq \mathcal{V}(C) \cup \widehat{\mathcal{V}}$.

Here, $\widehat{\mathcal{V}} = \{x_{k,j} \mid 1 \leq k \leq d-1 \text{ and } 1 \leq j \leq m_k\}$.

Since $s\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle = s_1\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle\sigma$ for some substitution σ that instantiates at most the $x_{i,j}$ for $i \neq d$ by terms from $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$, the statement of the lemma thus follows.

The statement (†) is proved by induction on $d - i$. In the base case, $i = d$ and $s_d\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle[\![\varphi]\!]$ is $C[g_1(r_1^*), \ldots, g_n(r_n^*)][\![\varphi]\!]$, i.e., it already has the required form.

In the step case, $i < d$ and the constrained term $s_i\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle[\![\varphi]\!]$ is identical to $f_i(y_i^*, s_{i+1}\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle, z_i^*)[\![\varphi]\!]$. The inductive hypothesis for $i+1$ implies that there exists a simplification tree for the constrained term $s_{i+1}\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle[\![\varphi]\!]$ such that all leaves in this tree have labels of the form

$$E[s_{i+1}\langle g_{j_1}(r_{j_1}^*)\rangle, \ldots, s_{i+1}\langle g_{j_l}(r_{j_l}^*)\rangle][\![\varphi \wedge \psi]\!]$$

for a repeated f_{i+1}-context $E[\![\psi]\!]$ with $\mathcal{V}(E) \subseteq \mathcal{V}(C) \cup \widehat{\mathcal{V}}$. Here, y_i^* abbreviates $x_{i,1}, \ldots, x_{i,j_i-1}$ and z_i^* abbreviates $x_{i,j_i+1}, \ldots, x_{i,m_i}$. Thus, there also exists a simplification tree for the constrained term $s_i\langle C[g_1(r_1^*), \ldots, g_n(r_n^*)]\rangle[\![\varphi]\!]$ such that all leaves in this tree have labels of the form

$$f_i(y_i^*, E[s_{i+1}\langle g_{j_1}(r_{j_1}^*)\rangle, \ldots, s_{i+1}\langle g_{j_l}(r_{j_l}^*)\rangle], z_i^*)[\![\varphi \wedge \psi]\!]$$

By Lemma 53, there exists a simplification tree for this constrained term such that all leaves in this tree have labels of the form

$$D[f_i(y_i^*, s_{i+1}\langle g_{d_1}(r_{d_1}^*)\rangle, z_i), \ldots, f_i(y_i^*, s_{i+1}\langle g_{d_e}(r_{d_e}^*)\rangle, z_i^*)][\![\varphi \wedge \psi \wedge \psi']\!]$$

for a repeated f_i-context $D[\![\psi']\!]$ such that $\mathcal{V}(D) \subseteq \mathcal{V}(E) \cup \widehat{\mathcal{V}} \subseteq \mathcal{V}(C) \cup \widehat{\mathcal{V}}$. Observing that these labels are of the required form $D[s_i\langle g_{d_1}(r_{d_1}^*)\rangle, \ldots, s_i\langle g_{d_e}(r_{d_e}^*)\rangle][\![\varphi \wedge \theta]\!]$ finishes the proof. □

Proof of Theorem 58. Let $D[f(x^*)] \equiv t$ be a simple nested conjecture and consider a rule $f(l^*) \to C[f(r_1^*), \ldots, f(r_m^*)][\![\varphi]\!] \in \mathcal{R}(f)$. Provided $f(x^*)$ and $f(l^*)$ are unifiable and $\varphi\sigma$ is LIA-satisfiable for $\sigma = \mathrm{mgu}(f(x^*), f(l^*))$, application of Expand to $D[f(x^*)] \equiv t$ produces (amongst others) the atomic conjecture

$D[C\sigma[f(r_1^*)\sigma, \ldots, f(r_m^*)\sigma]] \equiv t\sigma[\![\varphi\sigma]\!]$. After application of Expand, the set H of hypotheses consists of the rule $D[f(x^*)] \to t$.

By Lemma 55, there is a simplification tree for $D[C\sigma[f(r_1^*)\sigma, \ldots, f(r_m^*)\sigma]]$ such that all leaves of this tree have labels of the form

$$E[D[f(r_{d_1}^*)\sigma], \ldots, D[f(r_{d_e}^*)\sigma]][\![\psi]\!]$$

for some context E over $\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}$. This simplification tree can be built by suitable applications of $\mathsf{Case\text{-}Simplify}$.

Now, if $x_i = x_j$ for $i < j$, then $\langle i, j\rangle \in \mathcal{I}mp\mathcal{E}q(f)$. Since $x_i = x_j$ implies $l_i\sigma = l_j\sigma$, the definition of $\mathcal{I}mp\mathcal{E}q$ yields $r_{d_k,i}\sigma = r_{d_k,j}\sigma$ for all $1 \leq k \leq e$ similar to the reasoning in the proof of Theorem 42. Hence, $\mathsf{Simplify}$ applies e times to the above conjecture using the hypothesis $D[f(x^*)] \to t \in H$ to obtain $E[t\tau_1, \ldots, t\tau_e] \equiv t\sigma[\![\varphi\sigma \wedge \psi]\!]$, where $\tau_k = \{x^* \mapsto r_{d_k}^*\sigma\}$. Since both sides are from $\mathcal{T}(\mathcal{C}(\mathcal{R}) \cup \mathcal{F}_{\mathsf{LIA}}, \mathcal{V})$, either $\mathsf{Theory}_\top$ or $\mathsf{Theory}_\bot$ is applicable. □

Proof of Theorem 62. Adapt the proof of Theorem 58 in the same way the proof of Theorem 42 was adapted to obtain the proof of Theorem 48. □

References

1. Takahito Aoto. Dealing with non-orientable equations in rewriting induction. In Frank Pfenning, editor, *Proceedings of the 17th International Conference on Rewriting Techniques and Applications (RTA '06)*, volume 4098 of *Lecture Notes in Computer Science*, pages 242–256. Springer-Verlag, 2006.
2. Takahito Aoto. Soundness of rewriting induction based on an abstract principle. *IPSJ Digital Courier*, 4:58–68, 2008.
3. Alessandro Armando, Michaël Rusinowitch, and Sorin Stratulat. Incorporating decision procedures in implicit induction. *Journal of Symbolic Computation*, 34(4):241–258, 2002.
4. Franz Baader, editor. *Proceedings of the 19th International Conference on Automated Deduction (CADE '03)*, volume 2741 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2003.
5. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
6. Clark Barrett and Cesare Tinelli. CVC3. In Werner Damm and Holger Hermanns, editors, *Proceedings of the 19th International Conference on Computer Aided Verification (CAV '07)*, volume 4590 of *Lecture Notes in Computer Science*, pages 298–302. Springer-Verlag, 2007.
7. Adel Bouhoula. Automated theorem proving by test set induction. *Journal of Symbolic Computation*, 23(1):47–77, 1997.
8. Adel Bouhoula and Florent Jacquemard. Automated induction with constrained tree automata. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR '08)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 539–554. Springer-Verlag, 2008.
9. Robert S. Boyer and J Strother Moore. *A Computational Logic*. Academic Press, 1979.
10. Robert S. Boyer and J Strother Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. In *Machine Intelligence 11*, pages 83–124. Oxford University Press, 1988.
11. Alan Bundy. The automation of proof by mathematical induction. In J. Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 13, pages 845–911. Elsevier Science Publishers, 2001.
12. Alan Bundy, David Basin, Dieter Hutter, and Andrew Ireland. *Rippling: Meta-Level Guidance for Mathematical Reasoning*. Cambridge University Press, 2005.
13. Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for DPLL(T). In Thomas Ball and Robert Jones, editors, *Proceedings of the 18th Conference on Computer Aided Verification (CAV '06)*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94. Springer-Verlag, 2006.
14. Stephan Falke. *Term Rewriting with Built-In Numbers and Collection Data Structures*. PhD thesis, University of New Mexico, Albuquerque, NM, USA, 2009.
15. Stephan Falke and Deepak Kapur. Inductive decidability using implicit induction. In Miki Hermann and Andrei Voronkov, editors, *Proceedings of the 13th Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '06)*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 45–59. Springer-Verlag, 2006.
16. Laurent Fribourg. A strong restriction of the inductive completion procedure. *Journal of Symbolic Computation*, 8(3):253–276, 1989.

17. Jürgen Giesl and Deepak Kapur. Decidable classes of inductive theorems. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR '01)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 469–484. Springer-Verlag, 2001.
18. Jürgen Giesl and Deepak Kapur. Deciding inductive validity of equations. In Baader [4], pages 17–31.
19. John V. Guttag and James J. Horning. The algebraic specification of abstract data types. *Acta Informatica*, 10:27–52, 1978.
20. Christian Haselbach. Transformation techniques to verify imperative and functional programs. Diplomarbeit, Fachgruppe Informatik, Rheinisch-Westfälische Technische Hochschule Aachen, Germany, 2004.
21. Gérard P. Huet and Jean-Marie Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and Systems Sciences*, 25(2):239–266, 1982.
22. Dieter Hutter and Claus Sengler. INKA: The next generation. In Michael A. McRobbie and John K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE '96)*, volume 1104 of *Lecture Notes in Computer Science*, pages 288–292. Springer-Verlag, 1996.
23. Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4):1155–1194, 1986.
24. Jean-Pierre Jouannaud and Emmanuel Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1):1–33, 1989.
25. Deepak Kapur, Jürgen Giesl, and Mahadevan Subramaniam. Induction and decision procedures. *Revista de la Real Academia de Ciencias, Serie A: Matemáticas*, 98(1):153–180, 2004.
26. Deepak Kapur and David R. Musser. Proof by consistency. *Artificial Intelligence*, 31(2):125–157, 1987.
27. Deepak Kapur, Paliath Narendran, and Hantao Zhang. Automating inductionless induction using test sets. *Journal of Symbolic Computation*, 11(1–2):81–111, 1991.
28. Deepak Kapur and Mahadevan Subramaniam. New uses of linear arithmetic in automated theorem proving by induction. *Journal of Automated Reasoning*, 16(1–2):39–78, 1996.
29. Deepak Kapur and Mahadevan Subramaniam. Extending decision procedures with induction schemes. In David A. McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE '00)*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 324–345. Springer-Verlag, 2000.
30. Deepak Kapur and Hantao Zhang. An overview of Rewrite Rule Laboratory (RRL). *Computers & Mathematics with Applications*, 29(2):91–114, 1995.
31. Matt Kaufmann, Panagiotis Manolios, and J Strother Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, 2000.
32. Hirotaka Koike and Yoshihito Toyama. Inductionless induction and rewriting induction. *JSSST Computer Software*, 17(6):1–12, 2000. In Japanese.
33. David R. Musser. On proving inductive properties of abstract data types. In *Proceedings of the 7th ACM Symposium on Principles of Programming Languages (POPL '80)*, pages 154–162. ACM Press, 1980.
34. Uday S. Reddy. Term rewriting induction. In Mark E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction (CADE '90)*, volume 449 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 1990.
35. Barry K. Rosen. Tree-manipulating systems and church-rosser theorems. *Journal of the ACM*, 20(1):160–187, 1973.

36. Tsubasa Sakata, Naoki Nishida, Toshiki Sakabe, Masahiko Sakai, and Keiichirou Kusakari. Rewriting induction for constrained term rewriting systems. *IPSJ Transactions on Programming*, 2(2):80–96, 2009. In Japanese.
37. Sorin Stratulat. Combining rewriting with Noetherian induction to reason on non-orientable equalities. In Andrei Voronkov, editor, *Proceedings of the 19th International Conference on Rewriting Techniques and Applications (RTA '08)*, volume 5117 of *Lecture Notes in Computer Science*, pages 351–365. Springer-Verlag, 2008.
38. Christoph Walther and Stephan Schweitzer. About VeriFun. In Baader [4], pages 322–327.
39. Hantao Zhang, Deepak Kapur, and Mukkai S. Krishnamoorthy. A mechanizable induction principle for equational specifications. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the 9th Conference on Automated Deduction (CADE '88)*, volume 310 of *Lecture Notes in Computer Science*, pages 162–181. Springer-Verlag, 1988.