RWTH Aachen

Department of Computer Science
Technical Report

Deciding Inductive Validity of Equations

Jürgen Giesl and Deepak Kapur

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2003-3

RWTH Aachen · Department of Computer Science · July 2003 (revised version)

Deciding Inductive Validity of Equations[⋆⋆⋆]

Jürgen Giesl[1] and Deepak Kapur[2]

[1] LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
giesl@informatik.rwth-aachen.de

[2] Computer Science Dept., University of New Mexico, Albuquerque, NM 87131, USA
kapur@cs.unm.edu

Abstract. Kapur and Subramaniam [11] defined syntactical classes of equations where inductive validity can be decided automatically. However, these classes are quite restrictive, since defined function symbols with recursive definitions may only appear on one side of the equations. In this paper, we expand the decidable class of equations significantly by allowing both sides of equations to be expressed using defined function symbols. The definitions of these function symbols must satisfy certain restrictions which can be checked mechanically. These results are crucial to increase the applicability of decision procedures for induction.

1 Introduction

Mechanized induction often requires user interaction and is incomplete (provers fail for many valid conjectures). This is especially daunting to an application expert trying to use an induction prover in cases when conjectures are simple.

Recently, there has been a surge of interest in the role of decision procedures in tools for reasoning about computations, especially because of the success of BDD-based tools and model checkers in hardware verification. However, because of the above-mentioned challenges in automating induction proofs, such tools lack support for inductive reasoning on recursively defined data structures.

In [11], Kapur and Subramaniam proposed a methodology for integrating induction with decision procedures. In this way, they defined a syntactical class of equations where inductive validity is decidable. For example, an induction prover like *RRL* [9, 10, 14] using the cover set method is guaranteed to terminate with a "yes" or "no" answer on equations in this class. Similar statements also hold for other inductive theorem provers, e.g., *NQTHM* [4], *ACL-2* [12], *CLAM* [5, 6], *INKA* [1, 13], *SPIKE* [3]. In [8], these results are extended to quantifier-free formulas built from such equations. However, the class of equations defined in [11] is quite restrictive, since defined function symbols (i.e., functions defined by algorithms) may only appear on certain positions in one side of the equations.

Example 1 Let $\mathcal{T}_C$ be the theory of the free constructors $\mathsf{0}, \mathsf{s}$ for natural numbers and $\mathsf{nil}, \mathsf{cons}$ for linear lists. We regard the algorithms "+", min, dbl, len, and app.

[⋆] Extended version of a paper from the *Proceedings of the 19th International Conference on Automated Deduction (CADE-19)*, Miami, FL, USA, LNAI 2741, Springer-Verlag.

[⋆⋆] This research was partially supported by an NSF IT award CCR-0113611.

$$\begin{aligned}
\alpha_1^{+}: &\quad \mathsf{0} + y \to y\\
\alpha_2^{+}: &\quad \mathsf{s}(x) + y \to \mathsf{s}(x+y)\\[1em]
\alpha_1^{\mathsf{dbl}}: &\quad \mathsf{dbl}(\mathsf{0}) \to \mathsf{0}\\
\alpha_2^{\mathsf{dbl}}: &\quad \mathsf{dbl}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{dbl}(x)))\\[1em]
\alpha_1^{\mathsf{min}}: &\quad \mathsf{min}(\mathsf{0}, y) \to \mathsf{0}\\
\alpha_2^{\mathsf{min}}: &\quad \mathsf{min}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{0}\\
\alpha_3^{\mathsf{min}}: &\quad \mathsf{min}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(\mathsf{min}(x,y))\\[1em]
\alpha_1^{\mathsf{len}}: &\quad \mathsf{len}(\mathsf{nil}) \to \mathsf{0}\\
\alpha_2^{\mathsf{len}}: &\quad \mathsf{len}(\mathsf{cons}(n,x)) \to \mathsf{s}(\mathsf{len}(x))\\[1em]
\alpha_1^{\mathsf{app}}: &\quad \mathsf{app}(\mathsf{nil}, y) \to y\\
\alpha_2^{\mathsf{app}}: &\quad \mathsf{app}(\mathsf{cons}(n,x), y) \to \mathsf{cons}(n, \mathsf{app}(x,y))
\end{aligned}$$

The following conjectures should be checked for inductive validity.

$$\mathsf{dbl}(u+v) = u + \mathsf{dbl}(v) \tag{1}$$

$$\mathsf{dbl}(u+v) = \mathsf{dbl}(u) + \mathsf{dbl}(v) \tag{2}$$

$$(u+v)+w = u+(v+w) \tag{3}$$

$$\mathsf{min}(u+v, u+w) = u + \mathsf{min}(v,w) \tag{4}$$

$$\mathsf{len}(\mathsf{app}(u,v)) = \mathsf{len}(u) + \mathsf{len}(v) \tag{5}$$

$$\mathsf{s}(\mathsf{len}(\mathsf{app}(u,v))) = \mathsf{len}(\mathsf{app}(u, \mathsf{cons}(n,v))) \tag{6}$$

Such equations are not permitted in [11], since both sides have defined symbols. The restrictions in [11] ensure that each subgoal generated in an induction proof attempt simplifies to a formula with function symbols from a decidable theory. Indeed, if one attempts to prove (1) by induction on u, then the formula $\mathsf{dbl}(\underline{x}+v) = \underline{x} + \mathsf{dbl}(v) \;\Rightarrow\; \mathsf{dbl}(\underline{\mathsf{s}(x)}+v) = \underline{\mathsf{s}(x)} + \mathsf{dbl}(v)$ in the induction step case simplifies to the following formula. It contains "+" and dbl, i.e., its symbols are not from the signature of the (decidable) theory of free constructors.

$$\mathsf{s}(\mathsf{s}(x + \mathsf{dbl}(v))) = \mathsf{s}(x + \mathsf{dbl}(v)) \tag{7}$$

Example 2 We consider the (decidable) theory $\mathcal{T}_{PA}$ of Presburger Arithmetic with constructors $\mathsf{0}$, $\mathsf{1}$, "+". Regard an algorithm "$*$" with the rules $\alpha_1^*:\ \mathsf{0}*y \to \mathsf{0}$ and $\alpha_2^*:\ (x+\mathsf{1})*y \to x*y+y$. We want to prove the distributivity law.

$$u*(v+w) = u*v + u*w \tag{8}$$

Again, a defined symbol "$*$" is on both sides of (8). In a proof by induction on u, the step case $\underline{x}*(v+w) = \underline{x}*v+\underline{x}*w \;\Rightarrow\; \underline{(x+\mathsf{1})}*(v+w) = \underline{(x+\mathsf{1})}*v+\underline{(x+\mathsf{1})}*w$ simplifies to a formula with "$*$" (i.e., it is not from the signature of $\mathcal{T}_{PA}$):

$$(x*v+x*w)+(v+w) = (x*v+v)+(x*w+w) \tag{9}$$

In this paper, the class of equations handled in [11] is extended by allowing arbitrary terms involving defined function symbols on arbitrary positions of both sides of an equation. The main idea is to develop criteria for *safe* generalizations of equations. As shown above, in a proof attempt by induction, the resulting equation (subgoal) may not be from the signature of a decidable theory since it includes defined function symbols. In that case, the equation is generalized by replacing subterms with defined root symbols by new variables. For example, the subgoal (7) can be generalized to an (invalid) formula over $\mathcal{T}_C$'s signature

$$\mathsf{s}(\mathsf{s}(z)) = \mathsf{s}(z) \tag{10}$$

by replacing $x + \mathsf{dbl}(v)$ with a new variable z. Similarly, Equation (9) is generalized to a valid formula of the decidable theory of Presburger Arithmetic.

$$(z_1 + z_2) + (v + w) = (z_1 + v) + (z_2 + w) \tag{11}$$

In general, such a generalization (i.e., the replacement of a subterm with defined symbols by a new variable) can transform an inductively valid formula into a formula that is not inductively valid. To obtain a decision procedure for inductive validity, we have to guarantee that all generalizations performed are *safe* (i.e, the original formula is inductively valid if *and only if* the generalized formula is inductively valid). To this end we develop a *no-theory* condition which is sufficient for safe generalizations. Essentially, this means that if a subterm satisfies the no-theory condition, then its replacement by a fresh variable does not change the inductive validity of the equation.

Our induction proof procedure proceeds by applying induction to the original equation and by simplifying the resulting induction formulas afterwards. During this simplification, one can also apply the induction hypotheses. Finally, the resulting proof obligations are generalized by replacing all remaining subterms with defined symbols by fresh variables. Then one uses the decision procedure of the underlying theory.

To guarantee that the final proof obligations that result from induction and simplification really are of the form where generalizations are safe, we have to determine the subterms with defined symbols which may appear in these proof obligations and require that they must satisfy the no-theory condition. After introducing the required notions in Sect. 2, in Sect. 3 we present a technique to estimate which subterms with defined symbols can occur in subgoals during an induction proof attempt (without actually performing the induction proof). The basic idea of the technique is to keep track of those positions in a term which may be propagated into the final proof obligations.

Then in Sect. 4, we present the no-theory condition, i.e., we define a syntactical class of terms where generalizations are safe. Thus, if the generalized subgoal is not inductively valid, then so is the original subgoal. For example, without performing the proof attempts of (1) or (8), our syntactic criteria ensure that all generalizations in their proofs will be equivalence-preserving. So the generalized subgoals (10) (resp. (11)) are inductively valid iff the original subgoals (7) (resp. (9)) are valid. With these results, in Sect. 5 we define a large class DEC of equational conjectures whose inductive validity can be decided. For instance, DEC contains equations like (1) – (6) and (8). Checking whether an equation belongs to DEC is easy and fast, since it mainly relies on pre-compiled information about the algorithms of defined functions.

2 Background

We use many-sorted first-order logic where "=" is the only predicate symbol and "=" is reflexive, symmetric, transitive, and congruent. For a signature $\mathcal{F}$ and an infinite set of variables $\mathcal{V}$ we denote the set of (well-typed) *terms over* $\mathcal{F}$ by $\mathcal{T}erms(\mathcal{F},\mathcal{V})$ and the set of ground terms by $\mathcal{T}erms(\mathcal{F})$. A theory $\mathcal{T}$ is given by a finite signature $\mathcal{F}_\mathcal{T}$ and a set of axioms (i.e., closed formulas) $AX_\mathcal{T}$ over the signature $\mathcal{F}_\mathcal{T}$. The theory $\mathcal{T}$ is defined to be the set of all closed formulas φ over $\mathcal{F}_\mathcal{T}$ such that $AX_\mathcal{T} \models \varphi$ (then we also say that φ is *valid*). Here, "$\models$" is the usual (semantic) first-order consequence relation. We often omit leading universal quantifiers and we write $s =_\mathcal{T} t$ as a shorthand for $AX_\mathcal{T} \models \forall \, \ldots \; s = t$.

For the *theory* $\mathcal{T}_C$ *of free constructors*, $AX_{\mathcal{T}_C}$ consists of the following formulas.

$$\begin{array}{ll}
\neg c(x_1,\ldots,x_n) = c'(y_1,\ldots,y_m) & \text{for all } c, c' \in \mathcal{F}_{\mathcal{T}_C} \text{ where } c \neq c' \\
c(x_1,\ldots,x_n) = c(y_1,\ldots,y_n) \;\Rightarrow & \\
\qquad\qquad x_1 = y_1 \wedge \ldots \wedge x_n = y_n & \text{for all } c \in \mathcal{F}_{\mathcal{T}_C} \\
\bigvee_{c \in \mathcal{F}_{\mathcal{T}_C}} \exists y_1,\ldots,y_n.\; x = c(y_1,\ldots,y_n) & \\
\neg\; (c_1(\ldots c_2(\ldots c_n(\ldots x \ldots)\ldots)\ldots) = x) & \text{for all sequences } c_1,\ldots,c_n \text{ with } c_i \in \mathcal{F}_{\mathcal{T}_C}
\end{array}$$

Note that the last type of axioms usually results in infinitely many formulas. Here, "…" in the arguments of c_i stands for pairwise different variables. The above axioms state that all different constructor ground terms denote different objects and that every object starts with a constructor (still there may be models whose carrier contains objects which do not correspond to constructor ground terms).

We use the following definition for the *theory* $\mathcal{T}_{PA}$ *of Presburger Arithmetic*: $\mathcal{F}_{\mathcal{T}_{PA}} = \{0, 1, +\}$ and $AX_{\mathcal{T}_{PA}}$ consists of the following formulas:

$$\begin{array}{rr}
(x+y)+z = x+(y+z) & \neg\,(1+x = 0) \\
x+y = y+x & x+y = x+z \Rightarrow y = z \\
0+y = y & x = 0 \,\vee\, \exists y.\; x = y+1
\end{array}$$

For $t \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}_{PA}},\mathcal{V})$ with $\mathcal{V}(t) = \{x_1,\ldots,x_m\}$, there exist $a_i \in \mathbb{N}$ such that $t =_{\mathcal{T}_{PA}} a_0 + a_1 \cdot x_1 + \ldots + a_m \cdot x_m$. Here, "$a \cdot x$" denotes the term $x + \ldots + x$ (a times) and "a_0" denotes $1 + \ldots + 1$ (a_0 times). We often write *flattened* terms (i.e., without parentheses) since "+" is associative and commutative. For $s =_{\mathcal{T}_{PA}} b_0 + b_1 \cdot x_1 + \ldots + b_m \cdot x_m$ and t as above, we have $s =_{\mathcal{T}_{PA}} t$ iff $a_0 = b_0, \ldots, a_m = b_m$.

A formula is *inductively* valid if all its ground instantiations are valid. In applications like program verification one is usually interested in inductive validity and not in validity, since one wants to know how a program behaves when started with actual data. This data corresponds to ground instantiations of the variables. We often write x^* to denote a tuple of pairwise different variables.

Definition 3 (Inductive Validity) *A universal formula $\forall x^*.\ \varphi$ is inductively valid in the theory $\mathcal{T}$ (denoted $AX_\mathcal{T} \models_{ind} \varphi$) iff $AX_\mathcal{T} \models \varphi\sigma$ for all ground substitutions σ, i.e., σ substitutes all variables of φ by ground terms of $\mathcal{T}erms(\mathcal{F}_\mathcal{T})$.*

In general, validity implies inductive validity, but not vice versa. We restrict ourselves to theories like $\mathcal{T}_C$ and $\mathcal{T}_{PA}$ which are decidable and inductively complete (i.e., inductive validity of an equation $r_1 = r_2$ (*over* $\mathcal{F}_\mathcal{T}$) also implies its validity, cf. e.g. [7]). Then inductive validity of $r_1 = r_2$ can be checked by a decision procedure for $\mathcal{T}$. Of course, validity and inductive validity do no longer coincide if we introduce additional function symbols which are defined by algorithms.

We use *term rewrite systems* (TRSs) over a signature $\mathcal{F} \supseteq \mathcal{F}_\mathcal{T}$ as our programming language [2] and require that all left-hand sides of rules have the form $f(s^*)$ for a tuple of terms s^* from $\mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ and $f \notin \mathcal{F}_\mathcal{T}$. Thus, all our TRSs are constructor systems. Let $\mathcal{F}_d = \mathcal{F} \setminus \mathcal{F}_\mathcal{T}$ denote the set of *defined symbols.*

In order to perform evaluations with the TRS $\mathcal{R}$ and the underlying theory $\mathcal{T}$, we use the concept of rewriting modulo a theory ($\to_{\mathcal{R}/\mathcal{T}}$). As usual, we define $s \to_{\mathcal{R}/\mathcal{T}} t$ iff there exist s' and t' such that $s =_\mathcal{T} s' \to_\mathcal{R} t' =_\mathcal{T} t$. To ensure that our class of equations DEC is decidable, we have to require that the rewrite relation $\to_{\mathcal{R}/\mathcal{T}}$ must also be decidable. In other words, for two terms s and t it must be decidable whether $s \to_{\mathcal{R}/\mathcal{T}} t$ holds. For example, $\to_{\mathcal{R}/\mathcal{T}}$ is decidable whenever $\mathcal{T}$ is a theory where $\mathcal{T}$-equivalence classes of terms are finite and computable.

We restrict ourselves to terminating, confluent, and sufficiently complete TRSs. A TRS $\mathcal{R}$ is *terminating* iff $\to_{\mathcal{R}/\mathcal{T}}$ is well founded, it is *confluent* if $\to_{\mathcal{R}/\mathcal{T}}$ is confluent, and it is *sufficiently complete* if for all (well-typed) ground terms $t \in \mathcal{T}erms(\mathcal{F})$ there exists a $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T})$ such that $t \to^*_{\mathcal{R}/\mathcal{T}} q$. The term q is called a *normal form* and is often denoted as $t{\downarrow_{\mathcal{R}/\mathcal{T}}}$. When regarding $\to^*_{\mathcal{R}/\mathcal{T}}$ and ${\downarrow_{\mathcal{R}/\mathcal{T}}}$, we usually do not distinguish between terms that are equal w.r.t. $=_\mathcal{T}$.

The rules in $\mathcal{R}$ are considered as equational axioms extending the underlying theory $\mathcal{T}$. This results in a new theory with the signature $\mathcal{F}$ and the axioms $AX_\mathcal{T} \cup \{l = r \mid l \to r \in \mathcal{R}\}$. To ease readability, we write $AX_\mathcal{T} \cup \mathcal{R}$ instead of $AX_\mathcal{T} \cup \{l = r \mid l \to r \in \mathcal{R}\}$. It turns out that this extension is *conservative*, i.e., it does not change inductive validity of equations over $\mathcal{F}_\mathcal{T}$.

Theorem 4 (Inductive Validity of Equations over $\mathcal{F}_\mathcal{T}$) *For all $r_1, r_2 \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, we have $AX_\mathcal{T} \models_{ind} r_1 = r_2$ iff $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} r_1 = r_2$.*

Proof. The "only if" direction is trivial. For the "if" direction, we assume $AX_\mathcal{T} \not\models_{ind} r_1 = r_2$. Since we are only interested in inductive validity, it suffices to regard Herbrand interpretations (i.e., interpretations I such that for every element a in the universe there is a ground term t with $I(t) = a$).

Let I be a Herbrand interpretation such that $I \models AX_{\mathcal{T}}$ and $I \not\models r_1 = r_2$. Due to sufficient completeness and confluence, for every ground term t there is a *unique* ground term $q \in \mathcal{T}\mathit{erms}(\mathcal{F}_{\mathcal{T}})$ (up to equivalence under $\mathcal{T}$) such that $t \downarrow_{\mathcal{R}/\mathcal{T}} = q$. Hence, since $\mathcal{R}$ is a constructor system, we can extend I to an interpretation of $\mathcal{F}$ such that $I(l\sigma) = I(l\sigma\downarrow_{\mathcal{R}/\mathcal{T}})$ for all left-hand sides of rules l and all ground substitutions σ. But then I is a also model of $\{l = r \mid l \to r \in \mathcal{R}\}$ which means $AX_{\mathcal{T}} \cup \mathcal{R} \not\models_{ind} r_1 = r_2$. □

Decision procedures for theories $\mathcal{T}$ are integrated in many theorem provers. In this paper, we extend such decision procedures in order to handle functions defined by recursive rewrite rules as well. More precisely, we give syntactic conditions for equations whose inductive validity w.r.t. $AX_{\mathcal{T}} \cup \mathcal{R}$ is decidable. These conditions ensure that the induction proof attempt will reduce the original equation to equations over the signature $\mathcal{F}_{\mathcal{T}}$ of the underlying theory $\mathcal{T}$. Then by Thm. 4, their inductive validity (over the extended theory of $\mathcal{T}$ and $\mathcal{R}$) can be decided by a decision procedure for $\mathcal{T}$.[1]

In a proof attempt, induction is usually performed on *inductive* positions, since rewriting can only move a context outwards if it is on an inductive position.

Definition 5 (Inductive Positions) *For $f \in \mathcal{F}_d$, a position i with $1 \leq i \leq arity(f)$ is* non-inductive *if for all f-rules $f(s_1, \ldots, s_m) \to C[f(t_1^1, \ldots, t_m^1), \ldots, f(t_1^n, \ldots, t_m^n)]$ where C is a context over $\mathcal{F}_{\mathcal{T}}$, we have $s_i \in \mathcal{V}$, $t_i^k = s_i$, and $s_i \notin \mathcal{V}(s_j) \cup \mathcal{V}(t_j^k)$ for all $j \neq i$ and all $1 \leq k \leq n$. Otherwise, the position is* inductive.

For "+", dbl, len, app (Ex. 1) and "$*$" (Ex. 2), only the first argument positions are inductive. Without loss of generality, we assume that for every function f, the arguments $1, \ldots, j$ are inductive and $j + 1, \ldots, arity(f)$ are non-inductive for some $0 \leq j \leq arity(f)$. We often write rules in the form $f(s^*, y^*) \to C[f(t_1^*, y^*), \ldots, f(t_n^*, y^*)]$ to denote that C is a context over $\mathcal{F}_{\mathcal{T}}$ and $s^*, t_1^*, \ldots, t_n^*$ are the arguments on f's inductive positions. Most induction provers generate schemes for induction proofs (*cover sets*) from function definitions [4, 6, 13, 14].

Definition 6 (Cover Set) *Let $f \in \mathcal{F}_d$. Its* cover set *is $\mathcal{C}_f = \{\langle s^*, \{t_1^*, \ldots, t_n^*\}\rangle \mid f(s^*, y^*) \to C[f(t_1^*, y^*), \ldots, f(t_n^*, y^*)] \in \mathcal{R}\}$.*

An *induction on f* transforms a conjecture $\varphi[x^*]$ with pairwise different variables x^* into the following induction formulas for every $\langle s^*, \{t_1^*, \ldots, t_n^*\}\rangle \in \mathcal{C}_f$.

$$\varphi[t_1^*] \wedge \ldots \wedge \varphi[t_n^*] \;\Rightarrow\; \varphi[s^*] \tag{12}$$

If all induction formulas (12) are inductively valid, then by Noetherian induction the original formula $\varphi[x^*]$ is also inductively valid. The induction relation

[1] Thm. 4 and the restriction to theories where inductive validity of equations over $\mathcal{F}_{\mathcal{T}}$ implies their validity are also fundamental for the method of [11], but they are not mentioned there.

corresponds to the recursion structure of f and its well-foundedness follows from termination of $\mathcal{R}$.

In this paper, we develop criteria for equations $r_1 = r_2$ such that inductive validity is decidable. They ensure that there is a cover set $\mathcal{C}$ such that for every $\langle s^*, \{t_1^*, \ldots, t_n^*\}\rangle \in \mathcal{C}$, the induction conclusion $r_1[s^*] = r_2[s^*]$ can be simplified to $C[r_1[t_{i_1}^*], \ldots, r_1[t_{i_k}^*]] = D[r_2[t_{j_1}^*], \ldots, r_2[t_{j_l}^*]]$ for contexts C, D and $i_1, \ldots, j_l \in \{1, \ldots, n\}$. Here, $r[s^*]$ denotes that the induction variables are instantiated with the terms s^*. Thus, one can then apply the induction hypotheses $r_1[t_i^*] = r_2[t_i^*]$ to replace all occurrences of r_1 in the left-hand side by r_2. In the resulting conjecture

$$C[r_2[t_{i_1}^*], \ldots, r_2[t_{i_k}^*]] = D[r_2[t_{j_1}^*], \ldots, r_2[t_{j_l}^*]], \tag{13}$$

all remaining terms with defined root symbol can be generalized to fresh variables. We introduce a technique to estimate which subterms of r_1 and r_2 with defined symbols may occur in (13) without actually performing this induction proof attempt. Moreover, we present conditions on these subterms which guarantee that this generalization is safe. Finally, the decision procedure of the underlying theory can be used to decide the validity of the resulting formulas.

3 Compatibility Among Function Definitions

Our criteria for decidable equations rely on the notion of compatibility between *$\mathcal{T}$-based functions.*

Definition 7 ($\mathcal{T}$-based Function [11]) *A function $f \in \mathcal{F}$ is* $\mathcal{T}$-based *iff $f \in \mathcal{F}_\mathcal{T}$ or if all rules $l \to r \in \mathcal{R}$ with $root(l) = f$ have the form $f(s^*) \to C[f(t_1^*), \ldots, f(t_n^*)]$, where $s^*, t_1^*, \ldots, t_n^*$ are from $\mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ and C is a context over $\mathcal{F}_\mathcal{T}$.*

For instance, all algorithms in Ex. 1 are $\mathcal{T}_C$-based and in Ex. 2, "$*$" is $\mathcal{T}_{PA}$-based.

We will require that equations must have *compatible* sequences of $\mathcal{T}$-based functions on both sides. A function g is compatible with f on argument j if in any term $g(\ldots, f(\ldots), \ldots)$, where f is on the j-th argument of g, every context created by rewriting f will move outside the term by rewriting g. So if f has a rule $\alpha:\ f(s^*, y^*) \to C[f(t_1^*, y^*), \ldots, f(t_n^*, y^*)]$ with $n \geq 0$, then rewriting f can create the context C. Compatibility means that

$$g(x_1, \ldots, x_{j-1}, C[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m) \tag{14}$$

for $x_1, \ldots, x_m, z_1, \ldots, z_n \in \mathcal{V}$ will rewrite (in several steps) to some term

$$D[\, g(x_1, ..., x_{j-1}, z_{i_1}, x_{j+1}, ..., x_m),\ \ldots,\ g(x_1, ..., x_{j-1}, z_{i_k}, x_{j+1}, ..., x_m)\,] \tag{15}$$

where $i_1, \ldots, i_k \in \{1, \ldots, n\}$ and D is a context over $\mathcal{F}_\mathcal{T}$. Hence, if induction on f is performed within a term of the form $g(\ldots f(\ldots) \ldots)$, then in the induction conclusion, the resulting term $g(\ldots f(s^* ...) \ldots)$ can be rewritten to a term $D'[\,g(\ldots f(t^*_{i_1}...)\ldots),\,\ldots,\,g(\ldots f(t^*_{i_k}...)\ldots)\,]$. Here, the induction hypotheses $g(\ldots f(t^*_i...)\ldots)$ occur within a context D' (where D' is an instantiation of D).

For any f-rule α, let $\mathit{Rule}_{g,f}(\alpha)$ be the set of those g-rules used to rewrite (14) to (15) and let $\mathit{Var}_{g,f}(\alpha) = \{i \mid x_i \text{ occurs in } D\}$.[2] We make these rules and variable positions explicit to estimate which subterms with defined symbols may occur in subgoals during induction proofs. The reason is that the original term $g(\ldots f(\ldots)\ldots)$ may have defined symbols on positions from $\mathit{Var}_{g,f}(\alpha)$. These will be propagated outwards to the context D' during the induction proof.

In Ex. 1, "+" is compatible with dbl on argument 1. For $\alpha^{\mathsf{dbl}}_1 : \mathsf{dbl}(\mathsf{0}) \to \mathsf{0}$, C is $\mathsf{0}$ (a context without holes), and $\mathsf{0} + x_2$ rewrites to x_2 using α^+_1, i.e., $D = x_2$, $\mathit{Rule}_{+,\mathsf{dbl}}(\alpha^{\mathsf{dbl}}_1) = \{\alpha^+_1\}$, and $\mathit{Var}_{+,\mathsf{dbl}}(\alpha^{\mathsf{dbl}}_1) = \{2\}$, since D contains the variable x_2. For $\alpha^{\mathsf{dbl}}_2 : \mathsf{dbl}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{dbl}(x)))$, C is $\mathsf{s}(\mathsf{s}(\Box))$ and $\mathsf{s}(\mathsf{s}(z_1)) + x_2$ rewrites to $\mathsf{s}(\mathsf{s}(z_1 + x_2))$ by the rule α^+_2, i.e., $D = \mathsf{s}(\mathsf{s}(\Box))$, $\mathit{Rule}_{+,\mathsf{dbl}}(\alpha^{\mathsf{dbl}}_2) = \{\alpha^+_2\}$, and $\mathit{Var}_{+,\mathsf{dbl}}(\alpha^{\mathsf{dbl}}_2) = \varnothing$. Similarly, the function "+" is compatible with min and len on the argument 1.

Now we check whether "+" is compatible with itself on argument 1. For $\alpha^+_2 : \mathsf{s}(x) + y \to \mathsf{s}(x + y)$, we have $C = \mathsf{s}(\Box)$ and $\mathsf{s}(z_1) + x_2$ rewrites to $\mathsf{s}(z_1 + x_2)$, i.e., $D = \mathsf{s}(\Box)$, $\mathit{Rule}_{+,+}(\alpha^+_2) = \{\alpha^+_2\}$, and $\mathit{Var}_{+,+}(\alpha^+_2) = \varnothing$. For $\alpha^+_1 : \mathsf{0} + y \to y$, we have $C = y$, but $y + x_2$ does not rewrite to a term D over $\mathcal{F}_\mathcal{T}$. In general, for compatibility of g with f on argument j, we now permit that the compatibility requirement may be violated for some non-recursive rules $\mathit{Exc}_{g,f}$ of f ("$\underline{\mathit{exc}}$eptions"). However, the set $\mathit{Exc}_{g,f}$ should be as small as possible, i.e., a rule α should only be in $\mathit{Exc}_{g,f}$ if (14) does not rewrite to (15). Then, "+" is compatible with itself on argument 1 and $\mathit{Exc}_{+,+} = \{\alpha^+_1\}$.

Definition 8 (Compatible Functions) *Let g, f be $\mathcal{T}$-based, $f \notin \mathcal{F}_\mathcal{T}$, and $1 \le j \le m = \mathit{arity}(g)$. We say that g* is compatible with f on argument j *iff for all rules $\alpha :\ f(s^*, y^*) \to C[f(t^*_1, y^*), \ldots, f(t^*_n, y^*)]$, either $n = 0$ and $\alpha \in \mathit{Exc}_{g,f}$, or*

$$\begin{array}{l} g(x_1, \ldots, x_{j-1}, C[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m) \quad \to^*_{\mathcal{R}/\mathcal{T}} \\ D[\,g(x_1, \ldots, x_{j-1},\, z_{i_1},\, x_{j+1}, \ldots, x_m),\, \ldots,\, g(x_1, \ldots, x_{j-1},\, z_{i_k},\, x_{j+1}, \ldots, x_m)\,] \end{array}$$

for a context D over $\mathcal{F}_\mathcal{T}$, $i_1, ..., i_k \in \{1, ..., n\}$, and $z_i \notin \mathcal{V}(D)$ for all i. Let $\mathit{Rule}_{g,f}(\alpha)$ be the set of rules used in this reduction and let $\mathit{Var}_{g,f}(\alpha) = \{i \mid x_i \in \mathcal{V}(D)\}$.

With exceptions, in Ex. 1 dbl is also compatible with "+" and len is compatible with app. Note that in Def. 8, g can also be a symbol of $\mathcal{F}_\mathcal{T}$. For instance, s is compatible with len. We obtain $C = \mathsf{0}$ and $D = \mathsf{s}(\mathsf{0})$ for α^{len}_1 and $C = D = \mathsf{s}(\Box)$ for α^{len}_2. So for both len-rules α, $\mathit{Rule}_{\mathsf{s},\mathsf{len}}(\alpha) = \varnothing$ and $\mathit{Var}_{\mathsf{s},\mathsf{len}}(\alpha) = \varnothing$. Similarly, in Ex. 2, "+" is compatible with "$*$" on argument 1 and on argument 2.

[2] For a $\mathcal{T}$-based function f, $\mathit{Rule}_{g,f}(\alpha)$ is unique if $\mathcal{R}$ is non-overlapping. Otherwise, $\mathit{Rule}_{g,f}(\alpha)$ may be any set of g-rules which suffice to rewrite (14) to (15). $\mathit{Rule}_{g,f}$ and $\mathit{Var}_{g,f}$ also depend on the position j of g where the f-term occurs. But to ease the presentation we write $\mathit{Rule}_{g,f}$ and $\mathit{Var}_{g,f}$ instead of $\mathit{Rule}^j_{g,f}$ and $\mathit{Var}^j_{g,f}$.

Now we show that if g is compatible with f, then g cannot only process the context produced by one single defining equation of f, but it can also deal with *repeated* f-contexts. Such contexts are produced when several recursive calls of f are performed after another.

Definition 9 (Repeated f-Contexts) *Let f be a $\mathcal{T}$-based function and let R_f be a set of f-rules from $\mathcal{R}$. A context C is an* f-context *from the rules R_f iff there exists a rule $f(s^*, y^*) \to C[f(t_1^*, y^*), \ldots, f(t_n^*, y^*)] \in R_f$ where C is a context over $\mathcal{F}_\mathcal{T}$. A context C is a* repeated f-context *from the rules R_f iff C is an f-context from the rules R_f or if there are repeated f-contexts $D, C_1, \ldots, C_m$ from the rules R_f such that $C = D[C_1, \ldots, C_m]$.*

The following lemma states that if g is compatible with f then it can process every repeated f-context into a repeated g-context provided that no rules of $Exc_{g,f}$ were used to create the repeated f-context. If the f-context was produced by the f-rules R_f, then the resulting g-context can be created by g-rules from $Rule_{g,f}(\alpha)$ where $\alpha \in R_f$. Moreover, the variables in the g-context come from the original f-context or from g's argument positions $Var_{g,f}(\alpha)$.

Lemma 10 (Compatible Functions on Contexts) *Let g be compatible with f on argument j and let $Exc_{g,f} \cap R_f = \varnothing$. Then for every repeated f-context C_f resulting from the rules R_f there is a repeated g-context C_g resulting from the rules $\bigcup_{\alpha \in R_f} Rule_{g,f}(\alpha)$ such that*

$$\begin{array}{l} g(x_1, \ldots, x_{j-1}, C_f[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m) \quad \to^*_{\mathcal{R}/\mathcal{T}} \\ C_g[g(x_1, \ldots, x_{j-1}, z_{i_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{i_k}, x_{j+1}, \ldots, x_m)], \end{array}$$

where $i_1, \ldots, i_k \in \{1, \ldots, n\}$ and $\mathcal{V}(C_g) \subseteq \mathcal{V}(C_f) \cup \{x_i \mid i \in Var_{g,f}(\alpha),\ \alpha \in R_f\}$.

Proof. The lemma is proved by structural induction on C_f. If C_f is a non-repeated f-context, then there is a rule $\alpha :\ f(s^*, y^*) \to C_f[f(t_1^*, y^*), \ldots, f(t_n^*, y^*)] \in R_f$. By the definition of compatible functions (Def. 8) and since $Exc_{g,f} \cap R_f = \varnothing$, we have

$$\begin{array}{l} g(x_1, \ldots, x_{j-1}, C_f[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m) \ \to^*_{Rule_{g,f}(\alpha)/\mathcal{T}} \\ C_g[g(x_1, \ldots, x_{j-1}, z_{i_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{i_k}, x_{j+1}, \ldots, x_m)] \end{array}$$

with $i_1, \ldots, i_k \in \{1, \ldots, n\}$ and $\mathcal{V}(C_g) \subseteq \mathcal{V}(C_f) \cup \{x_i \mid i \in Var_{g,f}(\alpha)\}$. It remains to show that C_g is a repeated g-context.

Note that every term of the form $C[g(r_1^*), \ldots, g(r_k^*)]$, where C is a repeated g-context and r_i^* only contain symbols from $\mathcal{F}_\mathcal{T}$ can only rewrite to terms of the same form (since g is $\mathcal{T}$-based). Recall that C_f only contains symbols from $\mathcal{F}_\mathcal{T}$. So $g(x_1, \ldots, x_{j-1}, C_f[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m)$ is already of this form. Therefore, $C_g[g(\ldots), \ldots, g(\ldots)]$ must also have that form which means that C_g is indeed a repeated g-context.

If C_f is a repeated f-context then $C_f = C[C_1, \ldots, C_k]$ for repeated f-contexts $C, C_1, \ldots, C_k$ resulting from the rules R_f. By the induction hypothesis, there exists a repeated g-context D resulting from the rules $\bigcup_{\alpha \in R_f} \mathit{Rule}_{g,f}(\alpha)$ with $\mathcal{V}(D) \subseteq \mathcal{V}(C) \cup \{x_i \mid i \in \mathit{Var}_{g,f}(\alpha), \alpha \in R_f\}$ such that

$$\begin{array}{ll} g(x_1, \ldots, x_{j-1}, C_f[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m) & = \\ g(x_1, \ldots, x_{j-1}, C[C_1, \ldots, C_k][z_1, \ldots, z_n], x_{j+1}, \ldots, x_m) & \to^*_{\mathcal{R}/\mathcal{T}} \\ D[g(x_1, \ldots, x_{j-1}, u_1, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, u_d, x_{j+1}, \ldots, x_m)] & \end{array}$$

where $u_l \in \{C_e[z_1, \ldots, z_n] \mid 1 \leq e \leq k\}$ for $1 \leq l \leq d$. By the induction hypothesis there also exist repeated g-contexts $D_1, \ldots, D_d$ from the rules $\bigcup_{\alpha \in R_f} \mathit{Rule}_{g,f}(\alpha)$ with $\mathcal{V}(D_l) \subseteq \mathcal{V}(C_1) \cup \ldots \cup \mathcal{V}(C_k) \cup \{x_i \mid i \in \mathit{Var}_{g,f}(\alpha), \alpha \in R_f\}$ such that

$$\begin{array}{l} g(x_1, \ldots, x_{j-1}, u_l, x_{j+1}, \ldots, x_m) \quad \to^*_{\mathcal{R}/\mathcal{T}} \\ D_l[g(x_1, \ldots, x_{j-1}, z_{l_1}, x_{j+1}, \ldots, x_m), \ldots, g(x_1, \ldots, x_{j-1}, z_{l_{k_l}}, x_{j+1}, \ldots, x_m)] \end{array}$$

with $l_1, \ldots, l_{k_l} \in \{1, \ldots, n\}$. Hence,

$$\begin{array}{l} g(x_1, \ldots, x_{j-1}, C_f[z_1, \ldots, z_n], x_{j+1}, \ldots, x_m) \quad \to^*_{\mathcal{R}/\mathcal{T}} \\ D[g(x_1, .., x_{j-1}, u_1, x_{j+1}, .., x_m), \ldots, g(x_1, .., x_{j-1}, u_d, x_{j+1}, .., x_m)] \quad \to^*_{\mathcal{R}/\mathcal{T}} \\ D[D_1, ..., D_d][g(x_1, .., x_{j-1}, z_{1_1}, x_{j+1}, .., x_m), \ldots, g(x_1, .., x_{j-1}, z_{d_{k_d}}, x_{j+1}, .., x_m)], \end{array}$$

i.e., $C_g = D[D_1, \ldots, D_d]$. □

The concept of compatibility can be extended to arbitrarily deep nestings. To this end we define the notion of a *compatibility sequence*. Regard a term

$$r \ := \ f_1(p_1^*,\ f_2(p_2^*,\ f_3(x^*, q_3^*),\ q_2^*),\ q_1^*),$$

where the pairwise different variables x^* on f_3's inductive positions do not occur in the terms $p_1^*, p_2^*, q_1^*, q_2^*, q_3^*$. Moreover, let $f_1(p_1^*, f_2(\ldots), q_1^*)\,|_{j_1} = f_2(\ldots)$ and $f_2(p_2^*, f_3(\ldots), q_2^*)\,|_{j_2} = f_3(\ldots)$. The definition of compatibility sequences should guarantee that if $\langle f_1, f_2, f_3 \rangle$ is a compatibility sequence on the arguments $\langle j_1, j_2 \rangle$, then in an induction on f_3, the resulting context would be propagated outside of r. Hence, we require that f_i must be compatible with f_{i+1} on argument j_i for all $i \in \{1, 2\}$. So in Equation (6), $\langle \mathsf{s}, \mathsf{len}, \mathsf{app} \rangle$ is a compatibility sequence on $\langle 1, 1 \rangle$ and $\mathsf{s}(\mathsf{len}(\mathsf{app}(u, v)))$ is a term that *has* this compatibility sequence with the induction variable u.

An induction on f_3 would instantiate x^* according to the left-hand sides of f_3-rules $\alpha:\ f_3(s^*, y^*) \to C[f_3(t_1^*, y^*), \ldots, f_3(t_n^*, y^*)]$. For any term r as above, it should be guaranteed that $r[s^*]$ reduces to a term of the form $E[r[t_{i_1}^*], \ldots, r[t_{i_k}^*]]$ for some context E. For an instantiation C' of C, we clearly have

$$\begin{array}{rcl} r[s^*] & = & f_1(p_1^*,\ f_2(p_2^*,\ f_3(s^*, q_3^*),\ q_2^*),\ q_1^*) \\ & & \to_{\mathcal{R}/\mathcal{T}} f_1(p_1^*,\ f_2(p_2^*,\ C'[f_3(t_1^*, q_3^*), \ldots, f_3(t_n^*, q_3^*)],\ q_2^*),\ q_1^*). \end{array}$$

Since f_2 is compatible with f_3, C' can be moved outside and turned into a new context D by rewriting f_2. But this is only possible if no f_3-rule α from Exc_{f_2,f_3} was used to create the context C'. Then, the above term rewrites to

$$f_1(p_1^*,\ D[\,f_2(p_2^*,\ f_3(t_{j_1}^*, q_3^*),\ q_2^*),\ \ldots,\ f_2(p_2^*,\ f_3(t_{j_l}^*, q_3^*),\ q_2^*)\,],\ q_1^*).$$

As f_1 is compatible with f_2, f_1-rules can move D outside into a new context E. But again, this is only possible if no f_2-rules from Exc_{f_1,f_2} were used to produce the context D. For every f_3-rule $\alpha \notin \mathit{Exc}_{f_2,f_3}$, the set $\mathit{Rule}_{f_2,f_3}(\alpha)$ contains those f_2-rules which were used to create context D. Hence, we must demand $\mathit{Exc}_{f_1,f_2} \cap \mathit{Rule}_{f_2,f_3}(\alpha) = \varnothing$ for all f_3-rules $\alpha \notin \mathit{Exc}_{f_2,f_3}$. In this case, one can apply f_1-rules to the above term and obtains $E[r[t_{i_1}^*], \ldots, r[t_{i_k}^*]]$, i.e.,

$$E[\ f_1(p_1^*,\ f_2(p_2^*,\ f_3(t_{i_1}^*, q_3^*),\ q_2^*),\ q_1^*),\ \ldots,\ f_1(p_1^*,\ f_2(p_2^*,\ f_3(t_{i_k}^*, q_3^*),\ q_2^*),\ q_1^*)\ ].$$

The f_1-rules used to create context E are in $\mathit{Rule}_{f_1,f_2,f_3}(\alpha) = \mathit{Rule}_{f_1,f_2}(\beta_1) \cup \ldots \cup \mathit{Rule}_{f_1,f_2}(\beta_c)$, where $\mathit{Rule}_{f_2,f_3}(\alpha) = \{\beta_1, \ldots, \beta_c\}$. Computing $\mathit{Rule}_{f_1,f_2,f_3}(\alpha)$ would be required for compatibility sequences of four function symbols $\langle f_0, f_1, f_2, f_3\rangle$. In a term of the form $f_0(p_0^*, f_1(\ldots), q_0^*)$, we would also have to demand $\mathit{Exc}_{f_0,f_1} \cap \mathit{Rule}_{f_1,f_2,f_3}(\alpha) = \varnothing$ for all f_3-rules $\alpha \notin \mathit{Exc}_{f_2,f_3}$ in order to guarantee that in an f_3-induction, all resulting contexts are propagated outwards. So in general, from $\mathit{Rule}_{f_1,f_2}(\alpha), \ldots, \mathit{Rule}_{f_{d-1},f_d}(\alpha)$ one can immediately compute the set $\mathit{Rule}_{f_1,\ldots,f_d}(\alpha)$. It contains those f_1-rules which are needed for rewriting if the innermost f_d-term is instantiated according to the f_d-rule α. In Ex. 1, $\mathit{Rule}_{\mathsf{s},\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = \varnothing$, since $\mathit{Rule}_{\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = \{\alpha_2^{\mathsf{len}}\}$ and $\mathit{Rule}_{\mathsf{s},\mathsf{len}}(\alpha_2^{\mathsf{len}}) = \varnothing$.

Using $\mathit{Var}_{f_1,f_2}(\alpha), \ldots, \mathit{Var}_{f_{d-1},f_d}(\alpha)$, we can define a set $\mathit{Pos}_{f_1,\ldots,f_d}(\alpha)$. It contains the positions of those subterms of the original term that can occur in subgoals during proof attempts. Knowing the positions of these subterms allows us to formulate conditions for their safe generalization in Sect. 4.

Let us construct the set $\mathit{Pos}_{f_1,f_2,f_3}(\alpha)$ for f_3-rules $\alpha \notin \mathit{Exc}_{f_2,f_3}$. It contains the positions of r's subterms which may appear in the context E. Assume that we already know the positions $\mathit{Pos}_{f_2,f_3}(\alpha)$ of subterms in $f_2(p_2^*, f_3(\ldots), q_2^*)$ which occur in D. So these subterms are $f_2(p_2^*, f_3(\ldots), q_2^*)\,|_\pi$ for all $\pi \in \mathit{Pos}_{f_2,f_3}(\alpha)$. These terms can also appear in the final context E. Since $f_2(p_2^*, f_3(\ldots), q_2^*) = r|_{j_1}$, a subterm at position π in $f_2(p_2^*, f_3(\ldots), q_2^*)$ is at position $j_1\,\pi$ in r. Thus, $\mathit{Pos}_{f_1,f_2,f_3}(\alpha)$ should contain the positions $j_1\,\pi$ for all $\pi \in \mathit{Pos}_{f_2,f_3}(\alpha)$. Moreover, for every f_2-rule $\beta \in \mathit{Rule}_{f_2,f_3}(\alpha)$ which was used to create context D, the subterms of r at positions $\mathit{Var}_{f_1,f_2}(\beta)$ may occur in the final context E as well. In Ex. 1, we have $\mathit{Pos}_{\mathsf{s},\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = \mathit{Var}_{\mathsf{s},\mathsf{len}}(\alpha_2^{\mathsf{len}}) \cup \{1\,\pi \mid \pi \in \mathit{Pos}_{\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}})\} = \varnothing$ (as $\mathit{Rule}_{\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = \{\alpha_2^{\mathsf{len}}\}$ and $\mathit{Pos}_{\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = \varnothing$).

Def. 11 defines compatibility sequences of arbitrary length. In particular, $\langle f\rangle$ is a singleton compatibility sequence for any $\mathcal{T}$-based $f \in \mathcal{F}_d$. Here, if $f(p_1, ..., p_m)$ is rewritten with a rule $\alpha : f(s_1, ..., s_m) \to C[f(...), ..., f(...)]$, the resulting context is produced by α itself (i.e., $\mathit{Rule}_f(\alpha) = \{\alpha\}$). Let i be a non-inductive position of f. A defined function symbol in p_i can only be propagated into the context if $\mathcal{V}(s_i) \cap \mathcal{V}(C) \neq \varnothing$. In Ex. 1, $\langle + \rangle$ is a compatibility sequence with $Pos_+(\alpha_2^+) = \varnothing$ and $Pos_+(\alpha_1^+) = \{2\}$, since in the first rule $\mathsf{0}+y \to y$, the second argument y is moved to the context.

Definition 11 (Compatibility Sequence) *Let $d \geq 1$, let $r \in \mathcal{T}erms(\mathcal{F}, \mathcal{V})$, and let $f_1, \ldots, f_d$ be $\mathcal{T}$-based functions with $f_d \notin \mathcal{F}_\mathcal{T}$. The sequence $\langle f_1, \ldots, f_d \rangle$ is a* compatibility sequence *on arguments $\langle j_1, \ldots, j_{d-1} \rangle$ and the term r* has *this compatibility sequence with pairwise different induction variables x^* iff*

- *f_i is compatible with f_{i+1} on argument j_i and*
  $Exc_{f_i,f_{i+1}} \cap Rule_{f_{i+1},\ldots,f_d}(\alpha) = \varnothing$,
  for all $1 \leq i \leq d-1$ and all f_d-rules $\alpha \notin Exc_{f_{d-1},f_d}$
- *$r = f_1(p_1^*,\ f_2(p_2^*,\ \ldots\ f_{d-1}(p_{d-1}^*,\ f_d(x^*, q_d^*),\ q_{d-1}^*)\ \ldots,\ q_2^*),\ q_1^*)$,*
  where x^* are variables on f_d's inductive positions which do not occur elsewhere in r, and $f_i(p_i^*,\ f_{i+1}(\ldots),\ q_i^*)\,|_{j_i} = f_{i+1}(\ldots)$ for all $1 \leq i \leq d-1$
- *$Rule_{f_d}(\alpha) = \{\alpha\}$ and $Pos_{f_d}(\alpha) = \{i \mid \mathcal{V}(s_i) \cap \mathcal{V}(C) \neq \varnothing, i$ non-inductive$\}$,*
  for all f_d-rules $\alpha :\ f_d(s_1, \ldots, s_m) \to C[\,f_d(\ldots),\ \ldots,\ f_d(\ldots)\,]$
- *$Rule_{f_i,..,f_d}(\alpha) = \bigcup_{\beta \in Rule_{f_{i+1},..,f_d}(\alpha)} Rule_{f_i,f_{i+1}}(\beta)$ and*
  $Pos_{f_i,\ldots,f_d}(\alpha) = \bigcup_{\beta \in Rule_{f_{i+1},..,f_d}(\alpha)} Var_{f_i,f_{i+1}}(\beta)$
  $\qquad \cup \{j_i\,\pi \mid \pi \in Pos_{f_{i+1},..,f_d}(\alpha)\}$,

  for all $1 \leq i \leq d-1$ and all f_d-rules $\alpha \notin Exc_{f_{d-1},f_d}$

Whether $\langle f_1, \ldots, f_d \rangle$ is a compatibility sequence depends only on which functions are compatible with each other. This information can be pre-compiled. Then, it can be decided quickly whether a particular term has a compatibility sequence. Compatibility sequences and the functions *Rule* and *Pos* can also be computed at compile-time (but of course, these sequences can be arbitrarily long, so they can also be computed by need and stored for later re-use).

Lemma 12 shows that for a term with the compatibility sequence $\langle f_1, \ldots, f_d \rangle$ one can perform induction on f_d, as all resulting contexts can be propagated outwards.

Lemma 12 (Simplifying Terms with Compatibility Sequences) *Let r be a term with the compatibility sequence $\langle f_1, \ldots, f_d \rangle$ on the arguments $\langle j_1, ..., j_{d-1} \rangle$. For every rule $\alpha :\ f_d(s^*, y^*) \to C[f_d(t_1^*, y^*), \ldots, f_d(t_n^*, y^*)] \notin Exc_{f_{d-1},f_d}$, we have $r[s^*] \to_{\mathcal{R}/\mathcal{T}}^* D[r[t_{i_1}^*], \ldots, r[t_{i_k}^*]]$ for some $i_1, \ldots, i_k \in \{1, \ldots, n\}$ and context D. In D, defined symbols only occur within terms from $\{r|_\pi \mid \pi \in Pos_{f_1,\ldots,f_d}(\alpha)\}$.*

Proof. Let $r_d = f_d(x_{d,1}, \ldots, x_{d,j}, \ldots, x_{d,m_d})$ and $r_i = f_i(x_{i,1}, \ldots, x_{i,j_i-1}, r_{i+1}, x_{i,j_i+1}, \ldots, x_{i,m_i})$ for $1 \leq i \leq d-1$. Here, $x_{d,1}, \ldots, x_{d,j}$ are the induction variables of r. We prove the following statement for all $i \in \{1,\dots,d\}$:

$$\begin{array}{l} r_i[s^*] \to^*_{\mathcal{R}/\mathcal{T}} D[r_i[t^*_{j_1}],\dots,r_i[t^*_{j_l}]] \text{ for some } j_1,\dots,j_l \in \{1,\dots,n\} \\ \text{and } D \text{ is a repeated } f_i\text{-context from the rules } \mathit{Rule}_{f_i,\dots,f_d}(\alpha) \text{ with} \\ \mathcal{V}(D) \subseteq \mathcal{V}(s^*) \cup \{r_i|_\pi \mid \pi \in \mathit{Pos}_{f_i,\dots,f_d}(\alpha)\}. \end{array} \qquad (16)$$

As $r = r_1\sigma$ for some substitution σ, this implies $r[s^*] \to^*_{\mathcal{R}/\mathcal{T}} D\sigma[r[t^*_{j_1}],\dots,r[t^*_{j_l}]]$. Since every repeated f_1-context D is a context over $\mathcal{F}_\mathcal{T}$, all defined symbols in $D\sigma$ are introduced by σ. Thus, they only occur on or below variable positions of D. Since σ does not instantiate the induction variables, the variables in s^* are not modified by σ. Thus, (16) implies the lemma, because it states that defined symbols can only occur within the terms $r|_\pi$ where $\pi \in \mathit{Pos}_{f_1,\dots,f_d}(\alpha)$.

We prove (16) by induction on $d - i$. In the base case $i = d$ we have

$$\begin{array}{rcl} r_d[s^*] & = & f_d(s^*, x_{d,j+1},\dots,x_{d,m_d}) \\ & \to_{\mathcal{R}/\mathcal{T}} & C[f_d(t^*_1, x_{d,j+1},\dots,x_{d,m_d}),\dots,f_d(t^*_n, x_{d,j+1},\dots,x_{d,m_d})] \end{array}$$

where C is an f_d-context resulting from $\mathit{Rule}_{f_d}(\alpha) = \{\alpha\}$. By definition, $\mathcal{V}(C) \subseteq \mathcal{V}(s^*) \cup \{x_{d,i} \mid i \in \mathit{Pos}_{f_d}(\alpha)\}$.

Now we regard the induction step case where $i < d$. Let y^*_i abbreviate $x_{i,1},\dots,x_{i,j_i-1}$ and let z^*_i abbreviate $x_{i,j_i+1},\dots,x_{i,m_i}$. Then the induction hypothesis about r_{i+1} implies

$$r_i[s^*] = f_i(y^*_i, r_{i+1}[s^*], z^*_i) \to^*_{\mathcal{R}/\mathcal{T}} f_i(y^*_i, C[r_{i+1}[t^*_{j_1}],\dots,r_{i+1}[t^*_{j_l}]], z^*_i)$$

where C is a repeated f_{i+1}-context resulting from the rules $\mathit{Rule}_{f_{i+1},\dots,f_d}(\alpha)$ with $\mathcal{V}(C) \subseteq \mathcal{V}(s^*) \cup \{r_{i+1}|_\pi \mid \pi \in \mathit{Pos}_{f_{i+1},\dots,f_d}(\alpha)\}$.

Since $\mathit{Exc}_{f_i,f_{i+1}} \cap \mathit{Rule}_{f_{i+1},\dots,f_d}(\alpha) = \varnothing$, Lemma 10 implies that there must be a repeated f_i-context D resulting from the rules $\bigcup_{\beta \in \mathit{Rule}_{f_{i+1},\dots,f_d}(\alpha)} \mathit{Rule}_{f_i,f_{i+1}}(\beta) = \mathit{Rule}_{f_i,\dots,f_d}(\alpha)$ such that

$$\begin{array}{l} f_i(y^*_i, C[r_{i+1}[t^*_{j_1}],\dots,r_{i+1}[t^*_{j_l}]], z^*_i) \quad \to^*_{\mathcal{R}/\mathcal{T}} \\ D[f_i(y^*_i, r_{i+1}[t^*_{d_1}], z^*_i),\dots,f_i(y^*_i, r_{i+1}[t^*_{d_e}], z^*_i)] \end{array}$$

and

$$\begin{array}{rl} \mathcal{V}(D) & \subseteq \mathcal{V}(C) \cup \{r_i|_j \mid j \in \mathit{Var}_{f_i,f_{i+1}}(\beta), \beta \in \mathit{Rule}_{f_{i+1},\dots,f_d}(\alpha)\} \\ & \subseteq \mathcal{V}(s^*) \cup \{r_{i+1}|_\pi \mid \pi \in \mathit{Pos}_{f_{i+1},\dots,f_d}(\alpha)\} \\ & \qquad \cup \{r_i|_j \mid j \in \mathit{Var}_{f_i,f_{i+1}}(\beta), \beta \in \mathit{Rule}_{f_{i+1},\dots,f_d}(\alpha)\} \\ & = \mathcal{V}(s^*) \cup \{r_i|_\pi \mid \pi \in \mathit{Pos}_{f_i,\dots,f_d}(\alpha)\}. \end{array}$$

$\square$

In Ex. 1, let r be the term $u + \mathsf{dbl}(v)$. Then r has the compatibility sequence $\langle + \rangle$ on argument 1 with induction variable u. So on $+$'s non-inductive position 2 one may have terms like $\mathsf{dbl}(v)$ with defined symbols. The set *Pos* indicates which subterms may occur in the context of the simplified induction conclusion. Since $\mathit{Pos}_+(\alpha^+_1) = \{2\}$, $r|_2 = \mathsf{dbl}(v)$ can occur in the context when simplifying r.

Our notion of *compatibility* extends the one in [11] considerably. We extended compatibility by exceptions Exc and in a term $f_1(p_1^*, f_2(x^*, q_2^*), q_1^*)$ with a compatibility sequence $\langle f_1, f_2 \rangle$ and induction variables x^*, we permitted defined symbols in the terms p_1^*, q_1^*, q_2^*. Analogous statements hold for terms with longer compatibility sequences. For this reason, we had to introduce the sets Rule and Pos to trace which of the subterms with defined symbols are propagated outwards when rewriting f_1.

Moreover, in contrast to the "compatibility" in [11], we regard contexts C and D instead of single function symbols, we also permit symbols from $\mathcal{F}_\mathcal{T}$ to be compatible with other functions, and we allow permutations, duplications, and removal of recursive arguments in the definition of compatibility. (In Def. 8, the recursive calls $1, \ldots, n$ of f may be transformed into the calls $i_1, \ldots, i_k$, which may be arbitrary calls from $1, \ldots, n$.) With the notions of [11], the necessary compatibility requirements would not hold for the conjectures in Ex. 1 and Ex. 2. Indeed, the class of decidable equations recognized with our approach is a significant superset of the corresponding class in [11].

Finally, the requirement "$z_i \notin \mathcal{V}(D)$" in Def. 8 was erroneously missing in [11]. Without this requirement, simplification and applying the induction hypothesis does not necessarily result in an equation over $\mathcal{F}_\mathcal{T}$. A counterexample is a TRS containing the following rules (besides others) where $\mathsf{s}, \mathsf{c} \in \mathcal{F}_\mathcal{T}$.

$$\mathsf{g}(\mathsf{s}(x)) \to \mathsf{c}(x, \mathsf{g}(x))$$
$$\mathsf{f}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{f}(x))$$

If the requirement "$z_i \notin \mathcal{V}(D)$" would be dropped, then g would be compatible with f. However, if the induction conclusion contains the term $\mathsf{g}(\mathsf{f}(\mathsf{s}(x)))$, then it simplifies first to $\mathsf{g}(\mathsf{s}(\mathsf{f}(x)))$ and then to $\mathsf{c}(\mathsf{f}(x), \mathsf{g}(\mathsf{f}(x)))$. Replacing $\mathsf{g}(\mathsf{f}(x))$ by the other side of the induction hypothesis will still not result in a term over $\mathcal{F}_\mathcal{T}$ because of the remaining occurrence of $\mathsf{f}(x)$.

As in [11], the concept of compatibility can be extended to *simultaneous compatibility.* A binary function g is simultaneously compatible with f_1 and f_2 on argument positions 1 and 2, if f_1 and f_2 have the same cover set (up to variable renaming) and if g can simultaneously process the contexts C_1 and C_2 resulting from corresponding f_1- and f_2-rules. More precisely, we must have $f(C_1[y_1, \ldots, y_n], C_2[z_1, \ldots, z_n]) \to^*_{\mathcal{R}/\mathcal{T}} D[f(y_{i_1}, z_{i_1}), \ldots, f(y_{i_k}, z_{i_k})]$ for a context D over $\mathcal{F}_\mathcal{T}$. The general definition for simultaneous compatibility of functions g (of arbitrary arity) with arbitrary many functions $f_1, \ldots, f_m$ is analogous. Simultaneous compatibility can also be extended to arbitrarily deep nestings by defining corresponding compatibility sequences.

Of course, f_1 and f_2 may be identical. In Ex. 1, min is simultaneously compatible with "+" and "+" on the arguments 1 and 2 and thus $\langle \mathsf{min}, (+,+) \rangle$ is a simultaneous compatibility sequence. For α_2^+, we have $C_1 = C_2 = \mathsf{s}(\Box)$ and $\mathsf{min}(\mathsf{s}(y_1), \mathsf{s}(z_1)) \to \mathsf{min}(y_1, z_1)$, i.e., $D = \Box$. Thus, $\mathit{Rule}_{\mathsf{min},(+,+)}(\alpha_2^+) = \{\alpha_3^{\mathsf{min}}\}$, $\mathit{Pos}_{\mathsf{min},(+,+)}(\alpha_2^+) = \varnothing$, and $\mathit{Exc}_{\mathsf{min},(+,+)} = \{\alpha_1^+\}$. Moreover, in Ex. 2 the constructor "+" is simultaneously compatible with "$*$" and "$*$" on the arguments 1 and 2. To simplify the presentation, in the remainder we use a formulation with non-simultaneous compatibility in the definitions and theorems.

To guarantee[3] that the induction proof attempt for $r_1 = r_2$ transforms the equation into equivalent proof obligations over the theory $\mathcal{T}$, both r_1 and r_2 must have a compatibility sequence $\langle f_1, \ldots, f_d \rangle$ and $\langle g_1, \ldots, g_e \rangle$, respectively. (Alternatively, they may also be terms over $\mathcal{F}_\mathcal{T}$ which covers the equational conjectures discussed in [11].) If f_d and g_e have the same cover set (i.e., their recursion schemas correspond), then by compatibility, the context added on the arguments of f_d and g_e in induction conclusions will move outwards by rewriting. After application of the induction hypotheses, we obtain a proof obligation $C[t_1, \ldots, t_n] = D[s_1, \ldots, s_m]$ where C and D are contexts over $\mathcal{F}_\mathcal{T}$ and $t_1, \ldots, t_n, s_1, \ldots, s_m$ are subterms containing defined symbols. These subterms can already be determined before the induction proofs by inspecting the positions $Pos_{f_1,\ldots,f_d}(\alpha)$ and $Pos_{g_1,\ldots,g_e}(\alpha)$ of r_1 and r_2, respectively.

4 Safe Generalizations by the No-Theory Condition

To define the class of equations where inductive validity is decidable, we need syntactic criteria to ensure that an equation $C[t_1, ..., t_n] = D[s_1, ..., s_m]$ as above may be generalized to $C[x_{t_1}, ..., x_{t_n}] = D[x_{s_1}, ..., x_{s_m}]$. Here, t_i and s_j are replaced by fresh variables and identical terms are replaced by the same variable. This generalized equation is an equation over $\mathcal{F}_\mathcal{T}$ and thus, its (inductive) validity can be decided by a decision procedure for $\mathcal{T}$. In general, however, inductive validity of the generalized equation implies inductive validity of the original equation, but not vice versa. We define a *no-theory* condition which ensures that this generalization is *safe* in the theory of free constructors or Presburger Arithmetic.[4] Then an equation is inductively valid if *and only if* the generalized equation is inductively valid. Our condition mainly relies on information about the definitions of functions which can again be pre-compiled. A term satisfies the no-theory condition if it is not equivalent to any term without defined symbols.[5]

Definition 13 (No-Theory Condition) *A term t satisfies the no-theory condition iff there is no $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ with $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t = q$. If additionally, $t = f(x^*)$ for pairwise different variables x^*, we say that f satisfies the no-theory condition too.*

[3] Clearly, there are inductively valid equations where compatibility does not hold. Let half be defined by $\mathsf{half}(0) \to 0$, $\mathsf{half}(\mathsf{s}(0)) \to 0$, $\mathsf{half}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{half}(x))$. Then half is not compatible with "+" and thus, the conjecture $\mathsf{min}(\mathsf{half}(x), \mathsf{half}(x+y)) = \mathsf{half}(x)$ is not in our class *DEC* of equations where inductive validity is decidable.

[4] This criterion is generally applicable for safe generalizations, i.e., also outside of the framework of decidable induction proofs. Moreover, one could refine our approach by performing such generalizations also at the beginning before the start of the proof.

[5] Another criterion for safe generalizations would be to replace *surjective* terms t by fresh variables, i.e., terms t such that for all $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T})$ there is a substitution σ such that $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t\sigma = q$. However, this approach is very restrictive, since one would have to require that t's variables may not occur elsewhere in the formula.

Obviously, the no-theory condition is satisfied for almost all defined functions f (otherwise, the function f is not needed, since one can use the term q instead). For $\mathcal{T}_C$ and $\mathcal{T}_{PA}$, the no-theory condition for $\mathcal{T}$-based functions is decidable and we present syntactic sufficient conditions for the no-theory condition on terms.

If $f \in \mathcal{F}_d$ does not satisfy the no-theory condition, then there is a term $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ such that $q[x^*/s^*] =_\mathcal{T} r$ for every non-recursive f-rule $f(s^*) \to r$ (i.e., $r \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$). In the theory of free constructors, this means that $q[x^*/s^*]$ and r are syntactically identical. Thus, there are only finitely many possibilities for the choice of q. By checking whether these choices for q would contradict the remaining rules of f, we can decide the no-theory condition for f.

Definition 14 (Candidate Set $Q(f)$) *Let $\mathcal{T}$ be $\mathcal{T}_C$, let $f \in \mathcal{F}_d$ be a $\mathcal{T}$-based function of arity m. The candidate set $Q(f)$ is defined as $Q_{s^*}(r)$ for a non-recursive rule $f(s_1, \dots, s_m) \to r$. Let $x^* = x_1, \dots, x_m$ be pairwise different fresh variables not occurring in this rule. For any $t \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, we define $Q_{s^*}(t)$:*

$$\begin{aligned} Q_{s^*}(x) &= \{x_i \mid s_i = x\} && \text{for } x \in \mathcal{V}, \\ Q_{s^*}(c(t_1, \dots, t_k)) &= \{x_i \mid s_i = c(t_1, \dots, t_k)\} \cup \\ &\quad \{c(q_1, \dots, q_k) \mid q_i \in Q_{s^*}(t_i) \text{ for all } 1 \le i \le k\} && \text{for } c \in \mathcal{F}_\mathcal{T}. \end{aligned}$$

Now we show how to decide the no-theory condition for functions in the theory of free constructors.

Theorem 15 *Let $\mathcal{T}$, f be as in Def. 14. The function f satisfies the* no-theory condition *iff for every $q \in Q(f)$, there is an f-rule $l \to r$ with $l{\downarrow}_{f(x^*)\to q} \neq r{\downarrow}_{f(x^*)\to q}$. Here, $l{\downarrow}_{f(x^*)\to q}$ is the normal form of l w.r.t. the rule $f(x^*) \to q$.*

Proof. Let $s^*, t, q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ where s^* and t do not contain any of the variables x^* in q. Let $q[s^*]$ abbreviate $q[x^*/s^*]$. We first show that $q[s^*] =_\mathcal{T} t$ implies $q \in Q_{s^*}(t)$. Note that in the theory of free constructors, two terms from $\mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ can only be $=_\mathcal{T}$-equal if they are syntactically identical. Hence, we have to show that $q[s^*] = t$ implies $q \in Q_{s^*}(t)$. (In fact, the other direction holds as well, i.e., $q[s^*] =_\mathcal{T} t$ iff $q \in Q_{s^*}(t)$.)

We use induction on t. If $q[s^*] = x$ where $x \notin \mathcal{V}(q)$, then this implies $q = x_i$ and $s_i = x$ for some i. Hence, $q \in Q_{s^*}(x)$. If $q[s^*] = c(t_1, \dots, t_k)$, then there are again two possibilities. If q is a variable x_i, then we must have $s_i = c(t_1, \dots, t_k)$ for some i. Hence, $q \in Q_{s^*}(c(t_1, \dots, t_k))$. Otherwise, if $q \notin \mathcal{V}$, then $root(q)$ must be c. So q has the form $c(q_1, \dots, q_k)$ where $q_i[s^*] = t_i$. By the induction hypothesis this implies $q_i \in Q_{s^*}(t_i)$ and therefore $q \in Q_{s^*}(c(t_1, \dots, t_k))$.

Now we prove Thm. 15. For the “if” direction assume that f does not satisfy the no-theory condition, i.e., $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} f(x^*) = q$ holds for some term $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$. Thus, for f’s non-recursive rule $f(s^*) \to r$ which was chosen in the construction of $Q(f)$, we must have $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} r = q[s^*]$. Thus, $q[s^*] = r$ by Thm. 4 and the fact that we regard the theory of free constructors. According to the above argumentation this implies $q \in Q_{s^*}(r)$. So it suffices to regard only the terms q from $Q_{s^*}(r) = Q(f)$ when checking the no-theory condition.

By the assumption, there exists a rule $l \to r$ such that $l{\downarrow}_{f(x^*)\to q} \neq r{\downarrow}_{f(x^*)\to q}$. Since $l{\downarrow}_{f(x^*)\to q}$ and $r{\downarrow}_{f(x^*)\to q}$ are terms over $\mathcal{F}_\mathcal{T}$, in the theory of free constructors this implies $l{\downarrow}_{f(x^*)\to q} \neq_\mathcal{T} r{\downarrow}_{f(x^*)\to q}$ and by Thm. 4, $AX_\mathcal{T} \cup \mathcal{R} \not\models_{ind} l{\downarrow}_{f(x^*)\to q} = r{\downarrow}_{f(x^*)\to q}$. However, since $f(x^*) = q$ is inductively valid, this would mean $AX_\mathcal{T} \cup \mathcal{R} \not\models_{ind} l = r$ for a rule from $\mathcal{R}$ which is a contradiction.

For the "only if" direction, assume that $l{\downarrow}_{f(x^*)\to q} = r{\downarrow}_{f(x^*)\to q}$ holds for some $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$. We show that this implies $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} f(x^*) = q$ by induction w.r.t. the cover set C_f (where we do not distinguish between induction and non-induction variables). Hence, let $f(s^*) \to C[f(t_1^*), \ldots, f(t_n^*)]$ be a rule where C is a context over $\mathcal{F}_\mathcal{T}$. We have to prove inductive validity of the induction conclusion $f(s^*) = q[s^*]$, where we may assume inductive validity of $f(t_i^*) = q[t_i^*]$ for all i. Normalization of the induction conclusion results in $C[f(t_1^*), \ldots, f(t_n^*)] = q[s^*]$ and application of the induction hypothesis gives us the proof obligation $C[q[t_1^*], \ldots, q[t_n^*]] = q[s^*]$, i.e., $r{\downarrow}_{f(x^*)\to q} = l{\downarrow}_{f(x^*)\to q}$ for the rule $l \to r$ above. By the assumption, the terms on both sides of the equation are syntactically identical and hence, the proof obligation is inductively valid. $\square$

For "+" in Ex. 1, from the non-recursive rule $\mathsf{0} + y \to y$ we obtain $Q(+) = Q_{\mathsf{0},y}(y) = \{x_2\}$. However, the choice of $q = x_2$ contradicts the second rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$: normalizing by $x_1 + x_2 \to x_2$ produces non-identical terms y and $\mathsf{s}(y)$. Indeed, "+" (and also min, dbl, len, app) satisfy the no-theory condition.

For the theory of Presburger Arithmetic, if $f(x_1, \ldots, x_m) =_{\mathcal{T}_{PA}} q$ for a $q \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}_{PA}}, \mathcal{V})$, then $q =_{\mathcal{T}_{PA}} a_0 + a_1 \cdot x_1 + \ldots + a_m \cdot x_m$ for $a_i \in \mathbb{N}$ (see Sect. 1). We use the f-rules to compute constraints on the values of the coefficients a_i. Let τ map terms to linear polynomials where $\tau(x) = x$ for $x \in \mathcal{V}$, $\tau(\mathsf{0}) = 0$, $\tau(\mathsf{1}) = 1$, $\tau(s + t) = \tau(s) + \tau(t)$, and $\tau(f(t_1, \ldots, t_m)) = a_0 + \sum_{1 \le i \le m} a_i \cdot \tau(t_i)$. For every f-rule $l \to r$, we now require $\tau(l) = \tau(r)$. If $\mathcal{V}(l) = \{y_1, \ldots, y_k\}$, the polynomials $\tau(l) = P_0 + P_1 \cdot y_1 + \ldots + P_k \cdot y_k$ and $\tau(r) = Q_0 + Q_1 \cdot y_1 + \ldots + Q_k \cdot y_k$ are considered equal iff the constraints $P_0 = Q_0, \ldots, P_k = Q_k$ are satisfied. We generate such constraints for every f-rule. Since f is $\mathcal{T}$-based, its rules do not contain nested occurrences of f, and thus, P_i and Q_i are linear polynomials over $a_0, \ldots, a_m$. Thus, it is decidable whether the set of all these constraints is satisfiable. The constraints are unsatisfiable iff f satisfies the no-theory condition.

For "$*$" in Ex. 2, we assume that $x * y =_{\mathcal{T}_{PA}} a_0 + a_1 \cdot x + a_2 \cdot y$. The mapping τ is now applied to both defining equations of "$*$". From α_1^* we get $\tau(\mathsf{0} * y) = \tau(\mathsf{0})$, i.e., $a_0 + a_2 y = a_0$. From α_2^* we obtain $\tau((x + \mathsf{1}) * y) = \tau(x * y + y)$, i.e., $a_0 + a_1 + a_1 x + a_2 y = a_0 + a_1 x + (a_2 + 1) y$. Since polynomials are only considered equal if the corresponding coefficients are equal, the resulting set of constraints is $\{a_2 = 0, a_0 + a_1 = a_0, a_2 = a_2 + 1\}$ (plus trivial constraints). It is easy to detect their unsatisfiability and thus, "$*$" satisfies the no-theory condition.

We have described how to decide the no-theory condition for *functions*. Thm. 16 gives sufficient conditions for the no-theory condition on *terms*.

Theorem 16 *Let $\mathcal{T}$ be $\mathcal{T}_C$ or $\mathcal{T}_{PA}$. A term $t \in \mathcal{T}erms(\mathcal{F}, \mathcal{V})$ satisfies the no-theory condition if one of the following five conditions is satisfied:*

(a) $t = f(x^*)$ for pairwise different x^* and f satisfies the no-theory condition
(b) $t\sigma$ satisfies the no-theory condition for a substitution $\sigma : \mathcal{V} \to \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$
(c) $t \to^*_{\mathcal{R}/\mathcal{T}} r$ and r satisfies the no-theory condition
(d) $\mathcal{T} = \mathcal{T}_C$, $t|_\pi$ satisfies the no-theory condition, t has only $\mathcal{F}_\mathcal{T}$-symbols above π
(e) $\mathcal{T} = \mathcal{T}_{PA}$ and $t =_\mathcal{T} C[t_1, \ldots, t_n]$ for $n \geq 1$ and a context C over $\mathcal{F}_{\mathcal{T}_{PA}}$. Moreover, there is an $i \in \{1, \ldots, n\}$ such that t_i satisfies the no-theory condition and such that all t_j are either identical or variable disjoint to t_i.

Proof. Condition (a) is trivial since this holds by the definition of the no-theory condition for functions. For the remaining conditions, assume that t does not satisfy the no-theory condition, i.e., that $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t = q$ for some $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$.

(b) We also have $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t\sigma = q\sigma$ where $q\sigma \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, i.e., $t\sigma$ does not satisfy the no-theory condition either.
(c) Obviously, $t \to^*_{\mathcal{R}/\mathcal{T}} r$ implies $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t = r$. Therefore, we have $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} r = q$, i.e., r also violates the no-theory condition.
(d) In Condition (d), t has the form $C[t_1, \ldots, t_i, \ldots, t_n]$ where C is a context over $\mathcal{F}_\mathcal{T}$ and $t|_\pi = t_i$ for some $1 \leq i \leq n$, but where $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$ may be arbitrary terms from $\mathcal{T}erms(\mathcal{F}, \mathcal{V})$. In the theory of free constructors, $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} C[t_1, \ldots, t_i, \ldots, t_n] = q$ implies that $q = C[q_1, \ldots, q_i, \ldots, q_n]$ where $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t_j = q_j$ for all $1 \leq j \leq n$. Hence, we also have $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t_i = q_i$ in contradiction to the no-theory condition for $t_i = t|_\pi$.
(e) We first prove the following three statements for all $t, t_1, \ldots, t_n \in \mathcal{T}erms(\mathcal{F}, \mathcal{V})$ and all $s_1, \ldots, s_m \in \{\mathsf{0}, \mathsf{1}\} \cup \mathcal{V}$. So in contrast to $s_1, \ldots, s_m$, the terms $t, t_1, \ldots, t_n$ may contain defined symbols. Recall that in the theory of Presburger Arithmetic, every term $s \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ can be written in flattened form, where $s =_\mathcal{T} s_1 + \ldots + s_m$ for terms $s_1, \ldots, s_m \in \{\mathsf{0}, \mathsf{1}\} \cup \mathcal{V}$.

   (i) If $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} x + t_1 + \ldots + t_n = s_1 + \ldots + s_m$,
   then one of the s_i is x.
   (ii) If $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} \mathsf{1} + t_1 + \ldots + t_n = s_1 + \ldots + s_m$,
   then one of the s_i is $\mathsf{1}$.
   (iii) If $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t + \ldots + t = s_1 + \ldots + s_m$,
   then there exist $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, m\}$
   such that $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} t = s_{i_1} + \ldots + s_{i_k}$.

   To prove (i), assume that none of the s_i is x. Let s be the term $s_1 + \ldots + s_m$ and let $\sigma = \{x/s + \mathsf{1}\}$. This implies $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} s + \mathsf{1} + t_1\sigma + \ldots + t_n\sigma = s$. Since $AX_\mathcal{T}$ contains "$x + y = x + z \Rightarrow y = z$", we have $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} \mathsf{1} + t_1\sigma + \ldots + t_n\sigma = \mathsf{0}$ in contradiction to the axiom "$\neg\ \mathsf{1} + x = \mathsf{0}$".
   For (ii), assume that none of the s_i is $\mathsf{1}$. If σ is the substitution which replaces all occurring variables by $\mathsf{0}$, then this implies $s_i\sigma =_\mathcal{T} \mathsf{0}$ for all $i \in \{1, \ldots, m\}$.

Hence, we have $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} 1 + t_1\sigma + \ldots + t_n\sigma = \mathsf{0}$ in contradiction to the axiom "$\neg\ 1 + x = \mathsf{0}$".

For (iii), let "$t + \ldots + t$" consist of l additions of t. Let d be the number of those s_i's which are 1, i.e., $d = |\{i \mid 1 \leq i \leq m, s_i = 1\}|$. We claim that d is a multiple of l. To see this, let σ be a substitution which replaces all occurring variables by $\mathsf{0}$. By sufficient completeness, $t\sigma$ evaluates to a term from $\mathcal{T}erms(\mathcal{F}_{\mathcal{T}})$ and thus, $t =_{\mathcal{T}} 1 + \ldots + 1 = e \cdot 1$ for some $e \in \mathbb{N}$. Moreover, we have $s_1\sigma + \ldots + s_m\sigma =_{\mathcal{T}} d \cdot 1$. Thus, $t\sigma + \ldots + t\sigma =_{\mathcal{T}} (l \cdot e) \cdot 1 =_{\mathcal{T}} d \cdot 1$. The axioms "$x + y = x + z \Rightarrow y = z$" and "$\neg\ 1 + x = \mathsf{0}$" imply $l \cdot e = d$, i.e., d is a multiple of l.

For any variable x, let d_x be the number of those s_i's which are x, i.e., $d_x = |\{i \mid 1 \leq i \leq m, s_i = x\}|$. We claim that d_x is also a multiple of l. Let σ_x be a substitution which replaces x by 1 and all other occurring variables by $\mathsf{0}$. By sufficient completeness, $t\sigma_x$ evaluates to a term from $\mathcal{T}erms(\mathcal{F}_{\mathcal{T}})$, i.e., $t =_{\mathcal{T}} 1 + \ldots + 1 = e_x \cdot 1$ for some $e_x \in \mathbb{N}$. Moreover, $s_1\sigma_x + \ldots + s_m\sigma_x =_{\mathcal{T}} d_x \cdot 1$. Thus, we have $t\sigma_x + \ldots + t\sigma_x =_{\mathcal{T}} (l \cdot e_x) \cdot 1 =_{\mathcal{T}} d_x \cdot 1$. Again, the axioms "$x + y = x + z \Rightarrow y = z$" and "$\neg\ 1 + x = \mathsf{0}$" imply $l \cdot e_x = d_x$, i.e., d_x is a multiple of e_x.

Since the number of occurrences of every variable x and also of "1" in $s_1, \ldots, s_m$ is dividable by l, we can re-order $s_1, \ldots, s_m$ into l identical subsequences $s_{i_1}, \ldots, s_{i_k}$ (consisting of e occurrences of "1" and e_x occurrences of "x" for every variable x) plus a number of occurrences of $\mathsf{0}$. By the axiom "$x + y = x + z \Rightarrow y = z$", this implies $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} t = s_{i_1} + \ldots + s_{i_k}$.

Now we prove Condition (e). Assume that t does not satisfy the no-theory condition, i.e., $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1, \ldots, t_n] = q$ for some $q \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$. Let σ be a substitution which replaces all variables except those occurring in t_i by $\mathsf{0}$. By sufficient completeness, there exists a context C' over $\mathcal{F}_{\mathcal{T}}$ such that $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1, \ldots, t_n]\sigma = C'[t_i, \ldots, t_i]$ and by flattening terms, we result in a term $t' \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ such that $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C'[t_i, \ldots, t_i] = t' + t_i + \ldots + t_i$. Thus, $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} t' + t_i + \ldots + t_i = q\sigma$. Now (i) and (ii) imply that $q\sigma =_{\mathcal{T}} t' + q'$ for some term $q' \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$. By the axiom "$x + y = x + z \Rightarrow y = z$" we obtain $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} t_i + \ldots + t_i = q'$. But now (iii) implies that there exists a subterm q'' of q' such that $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} t_i = q''$ in contradiction to the no-theory condition for t_i. □

In $\mathcal{T}_C$, $\mathsf{dbl}(v)$ satisfies the no-theory condition since dbl satisfies the no-theory condition. Similarly, $\mathsf{s}(\mathsf{dbl}(v))$ satisfies the no-theory condition, since it only has the symbol $\mathsf{s} \in \mathcal{F}_{\mathcal{T}}$ above the no-theory term $\mathsf{dbl}(v)$. To benefit from Conditions (b) and (c), for example one can build all terms reachable from t by narrowing with non-recursive $\mathcal{T}$-based rules. (So termination is guaranteed, since the number of defined symbols decreases.) For instance, $x + \mathsf{dbl}(v)$ satisfies the no-theory condition, since it can be narrowed to $\mathsf{dbl}(v)$ with the non-recursive rule α_1^+. Similarly, $\mathsf{len}(u) + \mathsf{len}(v)$ can be narrowed to $\mathsf{len}(v)$ with the non-recursive rules α_1^{len} and α_1^+ and thus, it also satisfies the no-theory condition.

Condition (d) does not hold in the theory of Presburger Arithmetic. For example, let $\mathcal{R} = \{\mathsf{f}(0) \to 0,\ \mathsf{f}(x+1) \to x,\ \mathsf{g}(0) \to 0,\ \mathsf{g}(x+1) \to x+1+1\}$. Then $\mathsf{f}(x)$ and $\mathsf{g}(x)$ satisfy the no-theory condition, but $\mathsf{f}(x) + \mathsf{g}(x)$ does not, since $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} \mathsf{f}(x) + \mathsf{g}(x) = x + x$. However, in a term $C[t_1, \ldots, t_n]$ one may first apply a substitution σ (to unify non-variable disjoint terms t_i and t_j). If afterwards all remaining terms with defined symbols are variable disjoint from $t_i\sigma$ and if the term $t_i\sigma$ satisfies the no-theory condition, then this also holds for the original term. For example, $x * v + x * w$ satisfies the no-theory condition, because when instantiating v with w, then the instantiated term $x * w + x * w$ satisfies Condition (e).

Thm. 17 shows that the no-theory condition indeed allows us to replace pairwise variable disjoint terms by fresh variables. The "if" direction holds for arbitrary terms, but "only if" states that this never leads to "over-generalization".

Theorem 17 (Safe Generalization) *Let $\mathcal{T}$ be $\mathcal{T}_C$ or $\mathcal{T}_{PA}$ and let $t_1, \ldots, t_n$, $s_1, \ldots, s_m$ be pairwise identical or variable disjoint terms satisfying the no-theory condition. For all contexts C, D over $\mathcal{F}_{\mathcal{T}}$ and fresh variables x_{t_i} and x_{s_j}, we have $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1, ..., t_n] = D[s_1, ..., s_m]$ iff $C[x_{t_1}, \ldots, x_{t_n}] =_{\mathcal{T}} D[x_{s_1}, \ldots, x_{s_m}]$.*

Proof. The "if"-direction is trivial. We prove the "only if" direction. Assume that $C[x_{t_1}, \ldots, x_{t_n}] \neq_{\mathcal{T}} D[x_{s_1}, \ldots, x_{s_m}]$. We have to show that this contradicts the assumption

$$AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1, \ldots, t_n] = D[s_1, \ldots, s_m]. \tag{17}$$

Note that (17) implies $\{t_1, \ldots, t_n\} = \{s_1, \ldots, s_m\}$. Otherwise, without loss of generality, there exists an s_i with $s_i \notin \{t_1, \ldots, t_n\}$. Let σ be a substitution which replaces all variables from $(\mathcal{V}(t_1) \cup \ldots \cup \mathcal{V}(t_n) \cup \mathcal{V}(s_1) \cup \ldots \cup \mathcal{V}(s_m)) \setminus \mathcal{V}(s_i)$ with ground terms over $\mathcal{F}_{\mathcal{T}}$. By sufficient completeness and variable disjointness of the terms, there exists a $q \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ such that $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1, \ldots, t_n]\sigma = q$ and a context D' over $\mathcal{F}_{\mathcal{T}}$ with $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} D[s_1, \ldots, s_m] = D'[s_i, \ldots, s_i]$. Hence, (17) implies $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} D'[s_i, \ldots, s_i] = q$ which is a contradiction to the no-theory condition for s_i (by Thm. 16 (d) and (e), respectively).

First let $\mathcal{T}$ be the theory of free constructors. We perform structural induction on the contexts C and D. If $C = \Box$, then the equation in (17) has the form $t = D[t, \ldots, t]$ where D must contain $\Box$. If $D = \Box$, then we have $C[x_{t_1}, \ldots, x_{t_n}] = x_t = D[x_{s_1}, \ldots, x_{s_m}]$ in contradiction to the assumption $C[x_{t_1}, \ldots, x_{t_n}] \neq_{\mathcal{T}} D[x_{s_1}, \ldots, x_{s_m}]$. If $D \neq \Box$, then (17) contradicts an axiom "$\neg c_1(\ldots c_2(\ldots c_n(\ldots x \ldots) \ldots) \ldots) = x$" in $AX_{\mathcal{T}}$ which states that no term can be equal to one of its proper subterms.

Similar to the case $C = \Box$, the assumption $D = \Box$ also leads to a contradiction. So it remains to regard the case where $C \neq \Box$ and $D \neq \Box$. Hence, $C = c(C_1, \ldots, C_n)$ and the equation in (17) has the form $c(C_1, \ldots, C_n)[t_1, \ldots, t_n] = D[s_1, \ldots, s_m]$. If the equation would be inductively valid, then due to the fact that $D \neq \Box$ and that we regard the theory of free constructors, D would have the form $c(D_1,\dots,D_n)$ and all equations $C_i[t_1,\dots,t_n] = D_i[s_1,\dots,s_m]$ would be inductively valid. However, this is a contradiction to the induction hypotheses.

Now let $\mathcal{T}$ be the theory of Presburger Arithmetic. Let $t_{i_1},\dots,t_{i_k}$ be pairwise different terms such that $\{t_{i_1},\dots,t_{i_k}\} = \{t_1,\dots,t_n\}$. Since $\{t_1,\dots,t_n\} = \{s_1,\dots,s_m\}$, we have $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1,\dots,t_n] = a_1 \cdot t_{i_1} + \dots + a_k \cdot t_{i_k} + q_1$ and $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} D[s_1,\dots,s_m] = b_1 \cdot t_{i_1} + \dots + b_k \cdot t_{i_k} + q_2$ for terms $q_1, q_2 \in \mathit{Terms}(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ and $a_j, b_j \geq 1$.

We claim that $a_j = b_j$ for all j. To see this, let $j \in \{1,\dots,k\}$ be arbitrary and let σ be a substitution which replaces all variables from $\mathcal{V}(t_{i_1}) \cup \dots \cup \mathcal{V}(t_{i_{j-1}}) \cup \mathcal{V}(t_{i_{j+1}}) \cup \dots \cup \mathcal{V}(t_{i_k})$ by ground terms from $\mathit{Terms}(\mathcal{F}_{\mathcal{T}})$. By sufficient completeness and variable disjointness of the terms $t_{i_1},\dots,t_{i_k}$, we have $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1,\dots,t_n]\sigma = a_j \cdot t_{i_j} + q_1'$ and $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} D[s_1,\dots,s_m]\sigma = b_j \cdot t_{i_j} + q_2'$ for terms $q_1', q_2' \in \mathit{Terms}(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$. Without loss of generality let $a_j \geq b_j$. Then (17) and the axiom "$x+y = x+z \Rightarrow y = z$" imply $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} (a_j - b_j) \cdot t_{i_j} + q_1' = q_2'$. Here, "$(a_j - b_j) \cdot t_{i_j}$" stands for "$t_{i_j} + \dots + t_{i_j}$" where t_{i_j} is added $(a_j - b_j)$ times. If $a_j = b_j$, then it stands for "$\mathsf{0}$". By the observations (i) and (ii) from the proof of Thm. 16 (e), there exists a $q \in \mathit{Terms}(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ such that $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} q_2' = q + q_1'$ and hence, $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} (a_j - b_j) \cdot t_{i_j} = q$. By the observation (iii) from the proof of Thm. 16 (e), this is a contradiction to the no-theory condition of t unless $a_j = b_j$.

So we have proved $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[t_1,\dots,t_n] = a_1 \cdot t_{i_1} + \dots + a_k \cdot t_{i_k} + q_1$ and $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} D[s_1,\dots,s_m] = a_1 \cdot t_{i_1} + \dots + a_k \cdot t_{i_k} + q_2$. Now (17) and the axiom "$x + y = x + z \Rightarrow y = z$" imply $q_1 =_{\mathcal{T}} q_2$. However, this is a contradiction to the assumption $C[x_{t_1},\dots,x_{t_n}] \neq_{\mathcal{T}} D[x_{s_1},\dots,x_{s_m}]$, since $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} C[x_{t_1},\dots,x_{t_n}] = a_1 \cdot x_{t_{i_1}} + \dots + a_k \cdot x_{t_{i_k}} + q_1$ and $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} D[x_{s_1},\dots,x_{s_m}] = a_1 \cdot x_{t_{i_1}} + \dots + a_k \cdot x_{t_{i_k}} + q_2$. □

5 A Decidable Class of Equational Conjectures

Now we define the set *DEC* of equations whose inductive validity is decidable. Moreover, for any equation $r_1 = r_2$, it is decidable whether $r_1 = r_2 \in \mathit{DEC}$. Checking membership in *DEC* can be done efficiently, since it relies on precompiled information about compatibility and the no-theory condition of functions. Thus, before performing the induction proof one can recognize whether the equation will simplify to conjectures over the signature $\mathcal{F}_{\mathcal{T}}$ of the theory.

For $r_1 = r_2 \in \mathit{DEC}$, r_1 and r_2 must have compatibility sequences $\langle f_1,\dots,f_d\rangle$ and $\langle g_1,\dots,g_e\rangle$, where f_d and g_e have identical[6] cover sets (up to variable renaming). Then the induction conclusion can be simplified as described in Sect. 2.

The *Pos*-sets allow us to estimate which subterms of r_1 and r_2 with defined symbols will occur after this simplification without actually attempting an induction proof. Let $M(\alpha)$ denote the set of these subterms. Clearly, all $r_1|_\pi$ and $r_2|_{\pi'}$ for $\pi \in \mathit{Pos}_{f_1,\dots,f_d}(\alpha)$ and $\pi' \in \mathit{Pos}_{g_1,\dots,g_e}(\alpha)$ are in $M(\alpha)$. Moreover, the

[6] This requirement can be weakened by *merging* cover sets, cf. e.g. [4, 10, 13].

right-hand sides $r_2[t_1^*], \ldots, r_2[t_n^*]$ of induction hypotheses may also contain defined symbols. Finally, if $\alpha \in \mathit{Exc}_{f_{d-1},f_d}$, then compatibility does not hold for r_1. In this case, $M(\alpha)$ must include the whole simplified instantiated left-hand side r_1. A similar observation holds for the right-hand side r_2 if $\alpha \notin \mathit{Exc}_{g_{e-1},g_e}$. We require that all terms in $M(\alpha)$ with defined function symbols satisfy the no-theory condition. Then they can be safely generalized in induction proofs.

Definition 18 **(DEC)** *Let r_1, r_2 be terms in normal form. We define $r_1 = r_2 \in \mathit{DEC}$ iff r_1, r_2 are syntactically equal or if the following conditions are satisfied:*

- *$r_1 \in \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ or r_1 has a compatibility sequence $\langle f_1, \ldots, f_d\rangle$*
- *$r_2 \in \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ or r_2 has a compatibility sequence $\langle g_1, \ldots, g_e\rangle$*
- *If $r_1, r_2 \notin \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$, then the cover sets $\mathcal{C}_{f_d}$ and $\mathcal{C}_{g_e}$ are identical. Moreover, r_1 and r_2 have the same induction variables.*
- *If $r_1 \notin \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$, then for every f_d-rule α, terms in $M(\alpha) \setminus \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ are pairwise identical or variable disjoint and satisfy the no-theory condition.*

  Here, for $\alpha:\ f_d(s^*, y^*) \to C[f_d(t_1^*, y^*), \ldots, f_d(t_n^*, y^*)]$, α' is the corresponding[7] g_e-rule and $M(\alpha) = M_1(\alpha) \cup M_2(\alpha') \cup \{r_2[t_1^*], \ldots, r_2[t_n^*]\}$, where

$$M_1(\alpha) \;= \begin{cases} \{r_1|_\pi \mid \pi \in \mathit{Pos}_{f_1,\ldots,f_d}(\alpha)\} & \textit{if } \alpha \notin \mathit{Exc}_{f_{d-1},f_d} \\ \{r_1[s^*]\!\downarrow_{\mathcal{R}/\mathcal{T}}\} & \textit{if } \alpha \in \mathit{Exc}_{f_{d-1},f_d} \end{cases}$$

$$M_2(\alpha') = \begin{cases} \{r_2|_\pi \mid \pi \in \mathit{Pos}_{g_1,\ldots,g_e}(\alpha')\} & \textit{if } \alpha' \notin \mathit{Exc}_{g_{e-1},g_e} \\ \{r_2[s^*]\!\downarrow_{\mathcal{R}/\mathcal{T}}\} & \textit{if } \alpha' \in \mathit{Exc}_{g_{e-1},g_e} \end{cases}$$

For example, the equations (1), (2), (3), (5), (6) are in DEC. For the equation $\mathsf{dbl}(u + v) = u + \mathsf{dbl}(v)$, the left-hand side $\mathsf{dbl}(u + v)$ has the compatibility sequence $\langle \mathsf{dbl}, +\rangle$ and the right-hand side has the compatibility sequence $\langle +\rangle$ with the induction variable u. Since $\mathit{Exc}_{\mathsf{dbl},+} = \{\alpha_1^+\}$ and $\mathit{Pos}_+(\alpha_1^+) = \{2\}$, $M(\alpha_1^+)$ consists of $r_1[\mathsf{0}]\!\downarrow_{\mathcal{R}/\mathcal{T}} = \mathsf{dbl}(\mathsf{0} + v)\!\downarrow_{\mathcal{R}/\mathcal{T}} = \mathsf{dbl}(v)$ and of $r_2|_2 = \mathsf{dbl}(v)$. As $\mathit{Pos}_{\mathsf{dbl},+}(\alpha_2^+) = \mathit{Pos}_+(\alpha_2^+) = \varnothing$, $M(\alpha_2^+)$ only contains $r_2[x] = x + \mathsf{dbl}(v)$. The function dbl satisfies the no-theory condition and therefore, the terms $\mathsf{dbl}(v)$ and $x + \mathsf{dbl}(v)$ from $M(\alpha_1^+)$ and $M(\alpha_2^+)$ also fulfill the no-theory condition.

As mentioned in Sect. 3, compatibility may be extended to *simultaneous* compatibility and thus, this leads to a more general definition of DEC. Then, the equations (4) and (8) are also in DEC. For the distributivity equation $u*(v+w) = u*v+u*w$, the left-hand side has the compatibility sequence $\langle *\rangle$ and the right-hand side has the (simultaneous) sequence $\langle +, (*, *)\rangle$. Since $\mathit{Pos}_*(\alpha_1^*) =$

[7] Without loss of generality, $r_1 \notin \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ unless $r_1, r_2 \in \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$. If $r_2 \in \mathcal{T}\!\mathit{erms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ then $M_2(\ldots)$ is empty. Otherwise, for every f_d-rule α there is a *corresponding* g_e-rule α': $g_e(s^*, z^*) \to C'[g_e(t_1^*, z^*), \ldots, g_e(t_n^*, z^*)]$. We sometimes also write α instead of α'.

$Pos_{+,(*,*)}(\alpha_1^*) = \varnothing$, $Pos_*(\alpha_2^*) = \{2\}$, and $Pos_{+,(*,*)}(\alpha_2^*) = \{1\,2,\ 2\,2\}$, we obtain $M(\alpha_1^*) = \varnothing$, $M_1(\alpha_2^*) = \{v + w\}$, and $M_2(\alpha_2^*) = \{v, w\}$. So the only term with defined symbols in $M(\alpha_2^*)$ is $r_2[t^*]$, i.e., $x * v + x * w$. Our criteria in Thm. 16 state that this term satisfies the no-theory condition.

The following algorithm can decide inductive validity of all equations in DEC. Essentially, it uses cover set induction: First, the induction conclusions are normalized[8] and then the induction hypotheses are applied, if possible. Afterwards, all resulting proof obligations are generalized to equations over $\mathcal{F}_\mathcal{T}$. Finally, a decision procedure for $\mathcal{T}$ is applied to decide their validity. The induction proofs in Sect. 1 were performed in this way.

Algorithm $\mathsf{IND}(r_1, r_2)$

```
1. If $r_1$ and $r_2$ are syntactically identical then return "*True*".
2. If $r_1, r_2 \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, then use the decision procedure for $\mathcal{T}$ to
   decide the validity of $r_1 = r_2$ and return the respective result.
   Otherwise, without loss of generality, assume $r_1 \notin \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$.
3. Let $T$ consist of all subterms $f(\ldots)$ of $r_1$ which have pairwise different
   variables on the inductive positions of $f$.
4. If $T = \varnothing$ then stop and return "*False*".
5. Choose $f(\ldots) \in T$ and set $T = T \setminus \{f(\ldots)\}$.
6. For each $\langle s^*, \{t_1^*, \ldots, t_n^*\}\rangle \in \mathcal{C}_f$:
   6.1. Let $q_1 = r_1[s^*]\downarrow_{\mathcal{R}/\mathcal{T}}$ , $q_2 = r_2[s^*]\downarrow_{\mathcal{R}/\mathcal{T}}$ .
   6.2. Replace all occurrences of $r_1[t_i^*]$ in $q_1$ by $r_2[t_i^*]$.
   6.3. Replace all occurrences of subterms $t$ with $root(t) \in \mathcal{F}_d$
        in $q_1$ and $q_2$ by fresh variables $x_t$. So multiple occurrences of the same
        subterm are replaced by the same variable.
   6.4. Use the decision procedure for $\mathcal{T}$ to decide the validity of the
        resulting equation. If it is invalid, then go to Step 4.
7. Return "*True*".
```

In the definition of DEC we replace terms $t \in M(\alpha) \setminus \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ by new variables. In contrast in Step 6.3, only the subterms of t that have a defined root are replaced. For example, when proving the distributivity equation (8) we have $x{*}v{+}x{*}w \in M(\alpha)$, but in the algorithm the term $x{*}v{+}x{*}w$ would be replaced by $z_1 + z_2$ for new variables z_1 and z_2. Clearly, if this generalized conjecture is valid, then the original conjecture is valid, too. If the generalized conjecture is invalid, then the conjecture where the whole term $x * v + x * w$ would have been replaced by a new variable would also be invalid. Since DEC guarantees that even this (larger) generalization does not lead to over-generalization, the generalization in Step 6.3 is safe as well. Thus, one does not have to know about $M(\alpha)$ or DEC when performing induction proofs.

[8] If the induction hypotheses $r_1[t_i^*] = r_2[t_i^*]$ are not in normal form, then instead of reducing $r_1[s^*]$ and $r_2[s^*]$ to normal form in Step 6.1, one should stop as soon as $r_1[t_i^*]$ and $r_2[t_i^*]$ are reached.

The following theorem shows that IND is a decision procedure for all equations in DEC. Its proof can be found in the appendix.

Theorem 19 (Decision Procedure) *Let $\mathcal{T}$ be $\mathcal{T}_C$ or $\mathcal{T}_{PA}$, let $r_1 = r_2 \in DEC$. Then $\mathsf{IND}(r_1, r_2)$ terminates and it returns "True" iff $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} r_1 = r_2$. Hence, inductive validity is decidable for all equations in DEC.*

6 Conclusion and Further Work

The paper defines a syntactical class DEC of decidable equational conjectures by allowing defined function symbols to occur on both sides of an equation and also outside of inductive positions. This is a significant advance compared to earlier related work: In [11] only one side of an equation could have defined function symbols (only on inductive positions) and the other side had to be a term over the signature of the underlying decidable theory. In [8], we considered general quantifier-free conjectures with such equations as atomic formulas.

Our approach is based on compatibility between functions. Using this information, we identify those subterms which might appear in subgoals during a proof attempt and we require that these terms satisfy the no-theory condition. Then all subgoals can be safely generalized to formulas over a decidable theory.

Whether an equation belongs to our class DEC mainly depends on the definitions of functions. Therefore, the required information can be pre-compiled and checking whether an equation is in DEC can be done efficiently. Moreover, for every equation in DEC, a failed induction proof attempt refutes the conjecture. Thus, by restricting induction to this class of equations, one obtains a decision procedure for induction which can be integrated into fully automatic tools like model checkers or compilers.

In future work, we plan to relax the conditions imposed on function definitions further and to evaluate our approach empirically by an implementation. Moreover, we will try to extend our conditions for safe generalizations beyond the theories of free constructors and of Presburger Arithmetic. We also want to examine whether the ideas of [8] can be used to extend DEC to general quantifier-free conjectures whose atomic formulas are equations with defined symbols occurring on both sides. This class might be broadened further to include the use of intermediate lemmas in proofs, provided these lemmas themselves fall into the decidable class of inductively valid formulas.

Acknowledgments. We thank M. Subramaniam & R. Thiemann for helpful remarks.

References

1. S. Autexier, D. Hutter, H. Mantel, & A. Schairer. Inka 5.0 - A Logical Voyager. *Proc. CADE-16*, LNAI 1632, 1999
2. F. Baader & T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
3. A. Bouhoula & M. Rusinowitch. Implicit Induction in Conditional Theories. *Journal of Automated Reasoning*, 14:189–235, 1995.

4. R. S. Boyer and J S. Moore. *A Computational Logic*. Academic Press, 1979.
5. A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, & A. Smaill. Rippling: A Heuristic for Guiding Inductive Proofs. *Artificial Intelligence*, 62:185-253, 1993.
6. A. Bundy. The Automation of Proof by Mathematical Induction. A. Robinson & A. Voronkov (eds.), *Handbook of Automated Reasoning, Vol. 1*, pages 845-911, 2001.
7. H. B. Enderton. *A Mathematical Introduction to Logic*. 2nd edition, Harcourt/Academic Press, 2001.
8. J. Giesl & D. Kapur. Decidable Classes of Inductive Theorems. *Proc. IJCAR '01*, LNAI 2083, pages 469-484, 2001.
9. D. Kapur & H. Zhang. An Overview of Rewrite Rule Laboratory (*RRL*). *Journal of Computer and Mathematics with Applications*, 29:91–114, 1995.
10. D. Kapur & M. Subramaniam. New Uses of Linear Arithmetic in Automated Theorem Proving by Induction. *Journal of Automated Reasoning*, 16:39–78, 1996.
11. D. Kapur & M. Subramaniam. Extending Decision Procedures with Induction Schemes. *Proc. CADE-17*, LNAI 1831, pages 324-345, 2000.
12. M. Kaufmann, P. Manolios, & J S. Moore. *Computer-Aided Reasoning: An Approach*. Kluwer, 2000.
13. C. Walther. Mathematical Induction. D. M. Gabbay, C. J. Hogger, & J. A. Robinson (eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 2*, Oxford University Press, 1994.
14. H. Zhang, D. Kapur, & M. S. Krishnamoorthy. A Mechanizable Induction Principle for Equational Specifications. *Proc. CADE-9*, LNCS 310, 1988.

A Proof of Theorem 19

Theorem 19 (Decision Procedure) *Let $\mathcal{T}$ be $\mathcal{T}_C$ or $\mathcal{T}_{PA}$, let $r_1 = r_2 \in DEC$. Then $\mathsf{IND}(r_1, r_2)$ terminates and it returns "True" iff $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} r_1 = r_2$. Hence, inductive validity is decidable for all equations in DEC.*

Proof. Termination of the algorithm IND is obvious, i.e., the algorithm always returns "*True*" or "*False*". Hence, we now focus on the correctness of the algorithm IND.

If r_1 and r_2 are syntactically identical, then the claim is obvious. For two terms $r_1, r_2 \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ we have $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} r_1 = r_2$ iff $AX_{\mathcal{T}} \models_{ind} r_1 = r_2$ by Thm. 4. Since we restrict ourselves to theories where inductive validity of an equation over $\mathcal{F}_{\mathcal{T}}$ implies its validity, "$AX_{\mathcal{T}} \models_{ind} r_1 = r_2$" can be checked by the decision procedure for $\mathcal{T}$.

Otherwise, since $r_1 = r_2 \in DEC$, r_1 and r_2 have "suitable" compatibility sequences $\langle f_1, \ldots, f_d \rangle$ and $\langle g_1, \ldots, g_e \rangle$ (if $r_2 \notin \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$) which correspond to those in Def. 18. In the algorithm IND one also tries inductions which do not correspond to compatibility sequences (and inductions which correspond to other compatibility sequences). However, any induction proof is *sound*, i.e., if the algorithm returns "*True*", then the formula is really inductively valid. The reason is that the algorithm performs a sound Noetherian induction. In contrast, if a particular induction proof attempt fails, then this does not refute the formula, because of the generalization in Step 6.3. However, the algorithm IND only returns "*False*" if *all* induction proof attempts failed. Hence, then the attempt with the compatibility sequences from Def. 18 must have failed as well.

So it suffices to regard the proof attempt where the subterm "$f(\ldots)$" chosen in Step 5 is the subterm $f_d(\ldots)$ of the compatibility sequence in Def. 18. We have to show that for this proof attempt the algorithm IND only returns "*False*" if $r_1 = r_2$ is not inductively valid.

By the soundness of Noetherian induction and the sufficient completeness and termination of $\mathcal{R}$, $r_1 = r_2$ is inductively valid iff the conjectures

$$r_1[t_1^*] = r_2[t_1^*] \wedge \ldots \wedge r_1[t_n^*] = r_2[t_n^*] \;\Rightarrow\; r_1[s^*] = r_2[s^*] \tag{18}$$

are inductively valid for all rules

$$\alpha : \; f_d(s^*, y^*) \to C[f_d(t_1^*, y^*), \ldots, f_d(t_n^*, y^*)].$$

After Step 6.2, we obtain an equation $q_1 = q_2$ which results from (18) by rewriting $r_1[s^*]$ and $r_2[s^*]$ and by replacing occurrences of $r_1[t_i^*]$ by $r_2[t_i^*]$. Clearly, if $q_1 = q_2$ is inductively valid, then (18) is also inductively valid. So if we can verify $q_1 = q_2$ in Step 6.3 and 6.4 of the algorithm and proceed in a similar way for the other rules of f_d, then the algorithm *correctly* returns "*True*". On the other hand, if the original equation $r_1 = r_2$ is inductively valid, then $q_1 = q_2$ must also be inductively valid. Thus, if we can refute $q_1 = q_2$, then the algorithm *correctly* returns "*False*". So inductive validity of all these equations $q_1 = q_2$ is equivalent to inductive validity of $r_1 = r_2$.

It remains to prove that inductive validity of $q_1 = q_2$ is correctly determined in Step 6.3 and 6.4. To this end, we will now show that after Step 6.2, all subterms of $q_1 = q_2$ with defined root symbol occur in terms of $M(\alpha)$. Once this is shown, it is clear that the procedure in Step 6.3 and 6.4 indeed determines inductive validity of $q_1 = q_2$ correctly. The reason is that in Step 6.3 of the algorithm, only subterms of terms in $M(\alpha) \setminus \mathit{Terms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ are replaced by new variables, which results in an equation $s = t$ over $\mathcal{F}_\mathcal{T}$. Clearly, if $s =_\mathcal{T} t$, then $q_1 = q_2$ is valid, too. Otherwise, let $s' = t'$ be the equation which results from $q_1 = q_2$ by replacing all $t \in M(\alpha) \setminus \mathit{Terms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ by new variables x_t. Since the terms in $M(\alpha) \setminus \mathit{Terms}(\mathcal{F}_\mathcal{T}, \mathcal{V})$ are pairwise identical or variable disjoint and satisfy the no-theory-condition, by Thm. 17, $s' = t'$ is valid iff $q_1 = q_2$ is inductively valid. Note that $s = t$ is an instance of $s' = t'$. Thus, if $s = t$ is invalid, then $s' = t'$ is also invalid and thus, $q_1 = q_2$ is not inductively valid.

Now we show that after Step 6.2, all subterms of $q_1 = q_2$ with defined root symbol occur within terms of $M(\alpha)$.

<u>Case 1: $\alpha \notin \mathit{Exc}_{f_{d-1}, f_d}$</u>
Lemma 12 implies

$$r_1[s^*] \to^*_{\mathcal{R}/\mathcal{T}} C[r_1[t_{i_1}^*], \ldots, r_1[t_{i_k}^*]], \tag{19}$$

where $i_1, \ldots, i_k \in \{1, \ldots, n\}$ and in C, defined symbols only occur within terms from $\{r_1|_\pi \mid \pi \in \mathit{Pos}_{f_1,\ldots,f_d}(\alpha)\}$.

So in Step 6.1 of the algorithm we obtain $q_1 =_{\mathcal{T}} C[r_1[t^*_{i_1}], \dots, r_1[t^*_{i_k}]]$. By (19), Conjecture (18) is equivalent to

$$r_1[t^*_1] = r_2[t^*_1] \wedge \dots \wedge r_1[t^*_n] = r_2[t^*_n] \;\Rightarrow\; C[r_1[t^*_{i_1}], \dots, r_1[t^*_{i_k}]] = r_2[s^*]. \tag{20}$$

In Step 6.2 of the algorithm, this formula is transformed by applying the induction hypotheses. This results in

$$C[r_2[t^*_{i_1}], \dots, r_2[t^*_{i_k}]] = r_2[s^*], \tag{21}$$

i.e., $q_1 =_{\mathcal{T}} C[r_2[t^*_{i_1}], \dots, r_2[t^*_{i_k}]]$.

Case 1.1: $r_2 \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$

In this case, we also have $r_2[s^*], r_2[t^*_1], \dots, r_2[t^*_n] \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ and $q_2 =_{\mathcal{T}} r_2[s^*]$. Defined symbols in (21) can only occur within C. Thus, they all occur within terms of $\{r_1|_\pi \mid \pi \in Pos_{f_1,\dots,f_d}\} \subseteq M(\alpha)$. (Here, we have $M(\alpha) = \{r_1|_\pi \mid \pi \in Pos_{f_1,\dots,f_d}\} \cup \{r_2[t^*_1], \dots, r_2[t^*_n]\}$.)

Case 1.2: $r_2 \notin \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$, $\alpha' \notin Exc_{g_{e-1},g_e}$

In this case, in analogous way to r_1, by Lemma 12 we obtain

$$r_2[s^*] \to^*_{\mathcal{R}/\mathcal{T}} D[r_2[t^*_{j_1}], \dots, r_2[t^*_{j_l}]]$$

and in D, defined symbols only occur in terms from $\{r_2|_\pi \mid \pi \in Pos_{g_1,\dots,g_e}(\alpha')\}$. Thus, q_2 is $\mathcal{T}$-equivalent to $D[r_2[t^*_{j_1}], \dots, r_2[t^*_{j_l}]]$. Hence, all subterms of $q_1 = q_2$ with defined root symbol occur in terms of $M(\alpha) = \{r_1|_\pi \mid \pi \in Pos_{f_1,\dots,f_d}(\alpha)\} \cup \{r_2|_\pi \mid \pi \in Pos_{g_1,\dots,g_e}(\alpha')\} \cup \{r_2[t^*_1], \dots, r_2[t^*_n]\}$.

Case 1.3: $r_2 \notin \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$, $\alpha' \in Exc_{g_{e-1},g_e}$

Thus, α and α' are non-recursive rules. Therefore, $q_1 =_{\mathcal{T}} C$, $q_2 =_{\mathcal{T}} r_2[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}$, and there are no induction hypotheses. Hence, the subterms of $q_1 = q_2$ with defined root symbols occur in terms of $M(\alpha) = \{r_1|_\pi \mid \pi \in Pos_{f_1,\dots,f_d}\} \cup \{r_2[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}\}$.

Case 2: $\alpha \in Exc_{f_{d-1},f_d}$

Again, α is a non-recursive rule and $q_1 =_{\mathcal{T}} r_1[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}$.

Case 2.1: $r_2 \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$

In this case, we also have $r_2[s^*] \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ and thus defined symbols can only occur within terms of $M(\alpha) = \{r_1[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}\}$.

Case 2.2: $r_2 \notin \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$, $\alpha' \notin Exc_{g_{e-1},g_e}$

In this case, $q_2 =_{\mathcal{T}} D$, since α' is also non-recursive. Hence, all subterms of $q_1 = q_2$ with defined root symbol occur in terms of $M(\alpha) = \{r_1[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}\} \cup \{r_2|_\pi \mid \pi \in Pos_{g_1,\dots,g_e}(\alpha')\}$.

Case 2.3: $r_2 \notin \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$, $\alpha' \in Exc_{g_{e-1},g_e}$

Thus, $q_2 =_{\mathcal{T}} r_2[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}$ and the subterms of $q_1 = q_2$ with defined root symbols occur in terms of $M(\alpha) = \{r_1[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}\} \cup \{r_2[s^*]{\downarrow_{\mathcal{R}/\mathcal{T}}}\}$. $\square$

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

95-11 *	M. Staudt / K. von Thadden: Subsumption Checking in Knowledge Bases
95-12 *	G.V. Zemanek / H.W. Nissen / H. Hubert / M. Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
95-13 *	M. Staudt / M. Jarke: Incremental Maintenance of Externally Materialized Views
95-14 *	P. Peters / P. Szczurko / M. Jeusfeld: Business Process Oriented Information Management: Conceptual Models at Work
95-15 *	S. Rams / M. Jarke: Proceedings of the Fifth Annual Workshop on Information Technologies & Systems
95-16 *	W. Hans / St. Winkler / F. Sáenz: Distributed Execution in Functional Logic Programming
96-1 *	Jahresbericht 1995
96-2	M. Hanus / Chr. Prehofer: Higher-Order Narrowing with Definitional Trees
96-3 *	W. Scheufele / G. Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
96-4	K. Pohl: PRO-ART: Enabling Requirements Pre-Traceability
96-5	K. Pohl: Requirements Engineering: An Overview
96-6 *	M. Jarke / W. Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
96-7	O. Chitil: The ς-Semantics: A Comprehensive Semantics for Functional Programs
96-8 *	S. Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
96-9	M. Hanus (Ed.): Proceedings of the Poster Session of ALP'96 — Fifth International Conference on Algebraic and Logic Programming
96-10	R. Conradi / B. Westfechtel: Version Models for Software Configuration Management
96-11 *	C. Weise / D. Lenzkes: A Fast Decision Algorithm for Timed Refinement
96-12 *	R. Dömges / K. Pohl / M. Jarke / B. Lohmann / W. Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
96-13 *	K. Pohl / R. Klamma / K. Weidenhaupt / R. Dömges / P. Haumer / M. Jarke: A Framework for Process-Integrated Tools

96-14 * R. Gallersdörfer / K. Klabunde / A. Stolz / M. Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996

96-15 * H. Schimpe / M. Staudt: VAREX: An Environment for Validating and Refining Rule Bases

96-16 * M. Jarke / M. Gebhardt, S. Jacobs, H. Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

96-17 M. Jeusfeld / T. X. Bui: Decision Support Components on the Internet

96-18 M. Jeusfeld / M. Papazoglou: Information Brokering: Design, Search and Transformation

96-19 * P. Peters / M. Jarke: Simulating the impact of information flows in networked organizations

96-20 M. Jarke / P. Peters / M. Jeusfeld: Model-driven planning and design of cooperative information systems

96-21 * G. de Michelis / E. Dubois / M. Jarke / F. Matthes / J. Mylopoulos / K. Pohl / J. Schmidt / C. Woo / E. Yu: Cooperative information systems: a manifesto

96-22 * S. Jacobs / M. Gebhardt, S. Kethers, W. Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

96-23 * M. Gebhardt / S. Jacobs: Conflict Management in Design

97-01 Jahresbericht 1996

97-02 J. Faassen: Using full parallel Boltzmann Machines for Optimization

97-03 A. Winter / A. Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

97-04 M. Mohnen / S. Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

97-05 * S. Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

97-06 M. Nicola / M. Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

97-07 P. Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

97-08 D. Blostein / A. Schürr: Computing with Graphs and Graph Rewriting

97-09 C.-A. Krapp / B. Westfechtel: Feedback Handling in Dynamic Task Nets

97-10 M. Nicola / M. Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

97-13 M. Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

97-14 R. Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

97-15 G. H. Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

98-01 * Jahresbericht 1997

98-02	S. Gruner/ M. Nagel / A. Schürr: Fine-grained and Structure-oriented Integration Tools are Needed for Product Development Processes
98-03	S. Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr
98-04 *	O. Kubitz: Mobile Robots in Dynamic Environments
98-05	M. Leucker / St. Tobies: Truth — A Verification Platform for Distributed Systems
98-07	M. Arnold / M. Erdmann / M. Glinz / P. Haumer / R. Knoll / B. Paech / K. Pohl / J. Ryser / R. Studer / K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects
98-08 *	H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen
98-09 *	Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien
98-10 *	M. Nicola / M. Jarke: Performance Modeling of Distributed and Replicated Databases
98-11 *	A. Schleicher / B. Westfechtel / D. Jäger: Modeling Dynamic Software Processes in UML
98-12 *	W. Appelt / M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web
98-13	K. Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation
99-01 *	Jahresbericht 1998
99-02 *	F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version
99-03 *	R. Gallersdörfer / M. Jarke / M. Nicola: The ADR Replication Manager
99-04	M. Alpuente / M. Hanus / S. Lucas / G. Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing
99-07	Th. Wilke: CTL+ is exponentially more succinct than CTL
99-08	O. Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures
2000-01 *	Jahresbericht 1999
2000-02	Jens Vöge / Marcin Jurdzinski: A Discrete Strategy Improvement Algorithm for Solving Parity Games
2000-04	Andreas Becks / Stefan Sklorz / Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach
2000-05	Mareike Schoop: Cooperative Document Management
2000-06	Mareike Schoop / Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling
2000-07 *	Markus Mohnen / Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08	Thomas Arts / Thomas Noll: Verifying Generic Erlang Client-Server Implementations
2001-01 *	Jahresbericht 2000
2001-02	Benedikt Bollig / Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03	Thierry Cachat: The power of one-letter rational languages
2001-04	Benedikt Bollig / Martin Leucker / Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05	Benedikt Bollig / Martin Leucker / Thomas Noll: Regular MSC Languages
2001-06	Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07	Martin Grohe / Stefan Wöhrle: An Existential Locality Theorem
2001-08	Mareike Schoop / James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09	Thomas Arts / Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10	Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11	Klaus Indermark / Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 *	Jahresbericht 2001
2002-02	Jürgen Giesl / Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03	Benedikt Bollig / Martin Leucker / Thomas Noll: Generalised Regular MSC Languages
2002-04	Jürgen Giesl / Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05	Horst Lichter / Thomas von der Maßen / Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06	Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07	Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08	Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09	Markus Mohnen: Interfaces with Default Implementations in Java
2002-10	Martin Leucker: Logics for Mazurkiewicz traces
2002-11	Jürgen Giesl / Hans Zantema: Liveness in Rewriting
2003-01 *	Jahresbericht 2002
2003-02	René Thiemann / Jürgen Giesl: Size-Change Termination for Term Rewriting

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.