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 [12] 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 [12], Kapur & 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* [10, 11, 15] 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* [13], *CLAM* [5, 6], *INKA* [1, 14], *SPIKE* [3]. In [8], these results are extended to quantifier-free formulas built from such equations. However, the class of equations defined in [12] 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 following algorithms and conjectures.

$$\alpha_1^+ : \quad \mathsf{0} + y \to y \qquad\qquad \alpha_1^{\mathsf{dbl}} : \quad \mathsf{dbl}(\mathsf{0}) \to \mathsf{0}$$
$$\alpha_2^+ : \mathsf{s}(x) + y \to \mathsf{s}(x + y) \qquad\qquad \alpha_2^{\mathsf{dbl}} : \mathsf{dbl}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{dbl}(x)))$$

* This research was partially supported by an NSF ITR award CCR-0113611.
** *Proceedings of the 19th International Conference on Automated Deduction (CADE-19)*, Miami, FL, USA, LNAI 2741, Springer-Verlag, 2003.

$$\alpha_1^{\mathsf{min}}: \quad \mathsf{min}(0,y) \to 0 \qquad \alpha_1^{\mathsf{len}}: \quad \mathsf{len}(\mathsf{nil}) \to 0$$
$$\alpha_2^{\mathsf{min}}: \quad \mathsf{min}(\mathsf{s}(x),0) \to 0 \qquad \alpha_2^{\mathsf{len}}: \quad \mathsf{len}(\mathsf{cons}(n,x)) \to \mathsf{s}(\mathsf{len}(x))$$
$$\alpha_3^{\mathsf{min}}: \mathsf{min}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{s}(\mathsf{min}(x,y)) \qquad \alpha_1^{\mathsf{app}}: \quad \mathsf{app}(\mathsf{nil},y) \to y$$
$$\alpha_2^{\mathsf{app}}: \mathsf{app}(\mathsf{cons}(n,x),y) \to \mathsf{cons}(n,\mathsf{app}(x,y))$$

$$\mathsf{dbl}(u+v) = u + \mathsf{dbl}(v) \quad (1) \qquad \mathsf{min}(u+v,u+w) = u + \mathsf{min}(v,w) \quad (4)$$
$$\mathsf{dbl}(u+v) = \mathsf{dbl}(u) + \mathsf{dbl}(v) \quad (2) \qquad \mathsf{len}(\mathsf{app}(u,v)) = \mathsf{len}(u) + \mathsf{len}(v) \quad (5)$$
$$(u+v)+w = u+(v+w) \quad (3) \qquad \mathsf{s}(\mathsf{len}(\mathsf{app}(u,v))) = \mathsf{len}(\mathsf{app}(u,\mathsf{cons}(n,v))) \quad (6)$$

Such equations are not permitted in [12], since both sides have defined symbols. The restrictions in [12] 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.

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

Example 2. We consider the (decidable) theory $\mathcal{T}_{PA}$ of Presburger Arithmetic with constructors $0, 1$, "+". Regard an algorithm "$*$" with the rules $\alpha_1^*: \; 0{*}y \to 0$ and $\alpha_2^*: \; (x+1)*y \to x*y+y$. We want to prove the distributivity law (8).

$$u*(v+w) = u*v + u*w \quad (8)$$

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

$$(x*v + x*w) + (v+w) = (x*v+v) + (x*w+w) \quad (9)$$

In this paper, the class of equations handled in [12] 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

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

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.

$$(z_1+z_2)+(v+w) = (z_1+v)+(z_2+w) \quad (11)$$

In Sect. 2, we introduce required notions and sketch our overall approach. In Sect. 3, we present a technique to estimate which subterms with defined symbols occur in subgoals during an induction proof attempt (without actually performing the induction proof). Then in Sect. 4, we define a syntactical class of terms where generalizations are *safe*, i.e., 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 equations (containing (1) – (6) and (8)) whose inductive validity can be decided. Checking whether an equation belongs to DEC is fast, since it relies on pre-compiled information about defined functions. All proofs and further details can be found in [9].

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. Here, x^* denotes a tuple of pairwise different variables $x_1, \ldots, x_n$, etc.

$\neg c(x^*) = c'(y^*)$	for all $c, c' \in \mathcal{F}_{\mathcal{T}_C}$ where $c \neq c'$
$c(x_1,..,x_n) = c(y_1,..,y_n) \Rightarrow x_1 = y_1 \wedge ... \wedge x_n = y_n$	for all $c \in \mathcal{F}_{\mathcal{T}_C}$
$\bigvee_{c \in \mathcal{F}_{\mathcal{T}_C}} \exists y^*.\ x = c(y^*)$	
$\neg\ (c_1(\ldots c_2(\ldots c_n(\ldots x \ldots)\ldots)\ldots) = x)$	for all sequences $c_1, ..., c_n$, $c_i \in \mathcal{F}_{\mathcal{T}_C}$

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.

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 $a_0 = b_0, \ldots, a_m = b_m$. Instead of *validity*, we are usually interested in *inductive* validity.

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 com-

plete (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 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*.

To perform evaluations with the TRS $\mathcal{R}$ and the underlying theory $\mathcal{T}$, we use rewriting modulo a theory, where $\to_{\mathcal{R}/\mathcal{T}}$ must be decidable (e.g., this holds if $\mathcal{T}$-equivalence classes of terms are finite and computable). We have $s \to_{\mathcal{R}/\mathcal{T}} t$ iff there are s' and t' with $s =_\mathcal{T} s' \to_\mathcal{R} t' =_\mathcal{T} t$. We restrict ourselves to terminating, confluent, and sufficiently complete TRSs $\mathcal{R}$, where $\mathcal{R}$ 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 $t \to^*_{\mathcal{R}/\mathcal{T}} q$ (i.e., q is a *normal form* $t{\downarrow_{\mathcal{R}/\mathcal{T}}}$). 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 $AX_\mathcal{T} \cup \{l = r \,|\, l \to r \in \mathcal{R}\}$. To ease readability, we write $AX_\mathcal{T} \cup \mathcal{R}$ instead of $AX_\mathcal{T} \cup \{l = r \,|\, l \to r \in \mathcal{R}\}$. 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 $AX_\mathcal{T} \cup \mathcal{R} \models_{ind} r_1 = r_2$.*

Decision procedures for theories $\mathcal{T}$ are integrated in many theorem provers. In this paper, we extend 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 an induction proof attempt reduces 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 $\mathcal{T}$. In proofs, 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 $f(s_1, \ldots, s_m) \to C[f(t^1_1, \ldots, t^1_m), \ldots, f(t^n_1, \ldots, t^n_m)]$ where C is a context over $\mathcal{F}_\mathcal{T}$, we have $s_i \in \mathcal{V}$, $t^k_i = s_i$, and $s_i \notin \mathcal{V}(s_j) \cup \mathcal{V}(t^k_j)$ for all $j \neq i$ and $1 \leq k \leq n$. 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, 14, 15].

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

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 so is the original formula $\varphi[x^*]$ (by Noetherian induction). The induction relation 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 [12]). *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 $g(\ldots f(s^*\ldots)\ldots)$ can be rewritten to a term $D'[\, g(\ldots f(t^*_{i_1}\ldots)\ldots), \ldots, g(\ldots f(t^*_{i_k}\ldots)\ldots)\,]$. Here, the induction hypotheses $g(\ldots f(t^*_i\ldots)\ldots)$ 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 $\mathit{Var}_{g,f}(\alpha) = \{i \mid x_i \text{ occurs in } D\}$.[1] 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 $\alpha_1^{\mathsf{dbl}}: \mathsf{dbl}(\mathsf{0}) \to \mathsf{0}$, C is $\mathsf{0}$ (a context without holes), and $\mathsf{0}+x_2$ rewrites to x_2 using α_1^+, i.e., $D = x_2$, $\mathit{Rule}_{+,\mathsf{dbl}}(\alpha_1^{\mathsf{dbl}}) = \{\alpha_1^+\}$, $\mathit{Var}_{+,\mathsf{dbl}}(\alpha_1^{\mathsf{dbl}}) = \{2\}$, since D contains the variable x_2. For $\alpha_2^{\mathsf{dbl}}: \mathsf{dbl}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{dbl}(x)))$, 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 rule α_2^+, i.e., $D = \mathsf{s}(\mathsf{s}(\Box))$, $\mathit{Rule}_{+,\mathsf{dbl}}(\alpha_2^{\mathsf{dbl}}) = \{\alpha_2^+\}$, $\mathit{Var}_{+,\mathsf{dbl}}(\alpha_2^{\mathsf{dbl}}) = \varnothing$. Similarly, "+" is compatible with min and len on 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., $D = \mathsf{s}(\Box)$, $\mathit{Rule}_{+,+}(\alpha_2^+) = \{\alpha_2^+\}$, $\mathit{Var}_{+,+}(\alpha_2^+) = \varnothing$. 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 ("*exc*eptions"). However, 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 $1 \leq j \leq m = \mathit{arity}(g)$. 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*

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

for a context D over $\mathcal{F}_\mathcal{T}$, $i_1, ..., i_k \in \{1, ..., n\}$, $z_i \notin \mathcal{V}(D)$ 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, now 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_1^{len} and $C = D = \mathsf{s}(\Box)$ for α_2^{len}. 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.

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

[1] 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 $\mathit{Var}^j_{g,f}$.

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

where the pairwise different variables x^* on f_3's inductive positions do not occur in the terms p_i^*, q_j^*. Moreover, $f_1(p_1^*, f_2(...), q_1^*)\,|_{j_1} = f_2(...)$ and $f_2(p_2^*, f_3(...), q_2^*)\,|_{j_2} = f_3(...)$. 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 is 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

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

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 $Pos_{f_2,f_3}(\alpha)$ 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 $\pi \in Pos_{f_2,f_3}(\alpha)$. 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, $Pos_{f_1,f_2,f_3}(\alpha)$ should contain the positions $j_1\,\pi$ for all $\pi \in Pos_{f_2,f_3}(\alpha)$. Moreover, for every f_2-rule $\beta \in Rule_{f_2,f_3}(\alpha)$ which was used to create context D, the subterms of r at positions $Var_{f_1,f_2}(\beta)$ may occur in the final context E as well. In Ex. 1, we have $Pos_{\mathsf{s},\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = Var_{\mathsf{s},\mathsf{len}}(\alpha_2^{\mathsf{len}}) \cup \{1\,\pi \mid \pi \in Pos_{\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}})\} = \varnothing$ (as $Rule_{\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = \{\alpha_2^{\mathsf{len}}\}$ and $Pos_{\mathsf{len},\mathsf{app}}(\alpha_2^{\mathsf{app}}) = \varnothing$).

Def. 9 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., $Rule_f(\alpha) = \{\alpha\}$). 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 9 (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 $Pos_{f_d}(\alpha) = \{i \mid \mathcal{V}(s_i) \cap \mathcal{V}(C) \neq \varnothing, i \text{ non-inductive}\}$, for all f_d-rules $\alpha :\ f_d(s_1, \ldots, s_m) \to C[\,f_d(...),\ \ldots,\ f_d(...)\,]$*
- *$Rule_{f_i,\ldots,f_d}(\alpha) = \bigcup_{\beta \in Rule_{f_{i+1},\ldots,f_d}(\alpha)} Rule_{f_i,f_{i+1}}(\beta)$ and*
  $Pos_{f_i,\ldots,f_d}(\alpha) = \bigcup_{\beta \in Rule_{f_{i+1},\ldots,f_d}(\alpha)} Var_{f_i,f_{i+1}}(\beta) \cup \{j_i\,\pi \mid \pi \in Pos_{f_{i+1},\ldots,f_d}(\alpha)\}$,
  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 10 shows that for a term with the compatibility sequence $\langle f_1, \ldots, f_d \rangle$ one can do induction on f_d, as all resulting contexts can be propagated outwards.

Lemma 10 (Simplifying Terms with Compatibility Sequences). *Let r be a term with compatibility sequence $\langle f_1, \ldots, f_d \rangle$ on the arguments $\langle j_1, \ldots, j_{d-1} \rangle$. 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 $\{r|_\pi \mid \pi \in Pos_{f_1, \ldots, f_d}(\alpha)\}$.*

Our notion of *compatibility* extends the one in [12] considerably (see [9] for a detailed comparison). In particular, 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.

In Ex. 1, let r be the term $u + \mathsf{dbl}(v)$. Then r has the compatibility sequence $\langle + \rangle$ with induction variable u. So $+$ may have terms with defined symbols like $\mathsf{dbl}(v)$ on its non-inductive position 2. *Pos* indicates which subterms may occur in the context of the simplified induction conclusion. Since $Pos_+(\alpha_1^+) = \{2\}$, $r|_2 = \mathsf{dbl}(v)$ can occur in the context when simplifying r. Note that with the notions of [12], 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 [12].

As in [12], 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 g can simultaneously process the contexts C_1 and C_2 resulting from corresponding f_1- and f_2-rules. So we require $f(C_1[y_1, ..., y_n], C_2[z_1, ..., z_n]) \to^*_{\mathcal{R}/\mathcal{T}} D[f(y_{i_1}, z_{i_1}), \ldots, f(y_{i_k}, z_{i_k})]$ 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, $Rule_{\mathsf{min},(+,+)}(\alpha_2^+) = \{\alpha_3^{\mathsf{min}}\}$, $Pos_{\mathsf{min},(+,+)}(\alpha_2^+) = \varnothing$, $Exc_{\mathsf{min},(+,+)} = \{\alpha_1^+\}$. 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[2] 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 $\langle g_1, \ldots, g_e \rangle$ (alternatively, they

[2] Clearly, there are inductively valid equations where compatibility does not hold. Let half be defined by $\mathsf{half}(\mathsf{0}) \to \mathsf{0}$, $\mathsf{half}(\mathsf{s}(\mathsf{0})) \to \mathsf{0}$, $\mathsf{half}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{half}(x))$. 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.

may also be terms over $\mathcal{F}_\mathcal{T}$ which covers the equational conjectures discussed in [12]). 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.[3] 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.

Definition 11 (No-Theory). *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^*, then f satisfies the no-theory condition too.*

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 contradict the remaining rules of f, we can decide the no-theory condition for f.

Definition 12 (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 $f(s_1, \ldots, s_m) \to r$. Let $x^* = x_1, \ldots, x_m$ be pairwise different fresh*

[3] 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.

variables not occurring in this rule. For any $t \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$*, we define* $Q_{s^*}(t)$*:*

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

Theorem 13. *Let* $\mathcal{T}$*,* f *be as in Def. 12. 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* $f(x^*) \to q$*.*

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 $\tau((x+1) * y) = \tau(x * y + y)$, 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. 14 gives sufficient conditions for the no-theory condition on *terms*.

Theorem 14. *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* $n \ge 1$ *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*.*

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^+.

Condition (d) does not hold in the theory of Presburger Arithmetic. For example, let $\mathcal{R} = \{\mathsf{f}(\mathsf{0}) \to \mathsf{0},\ \mathsf{f}(x+\mathsf{1}) \to x,\ \mathsf{g}(\mathsf{0}) \to \mathsf{0},\ \mathsf{g}(x+\mathsf{1}) \to x+\mathsf{1}+\mathsf{1}\}$. 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. 15 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 15 (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 $C[x_{t_1}, \ldots, x_{t_n}] =_\mathcal{T} D[x_{s_1}, \ldots, x_{s_m}]$.*

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 $r_1 = r_2 \in DEC$. 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 $r_1 = r_2 \in DEC$, r_1 and r_2 must have compatibility sequences $\langle f_1, \ldots, f_d \rangle$ and $\langle g_1, \ldots, g_e \rangle$, where f_d and g_e have identical[4] 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 $\pi \in Pos_{f_1,\ldots,f_d}(\alpha)$ and $\pi' \in Pos_{g_1,\ldots,g_e}(\alpha)$ are in $M(\alpha)$. Moreover, the right-hand sides $r_2[t_1^*], \ldots, r_2[t_n^*]$ of induction hypotheses may also contain defined symbols. Finally, if $\alpha \in Exc_{f_{d-1},f_d}$, 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 $\alpha \notin Exc_{g_{e-1},g_e}$. 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.

[4] This requirement can be weakened by *merging* cover sets, cf. e.g. [4, 11, 14].

Definition 16 (DEC). *Let r_1, r_2 be terms in normal form. We define $r_1 = r_2 \in DEC$ iff r_1, r_2 are syntactically equal or the following conditions are satisfied:*

- *$r_1 \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ or r_1 has a compatibility sequence $\langle f_1, \ldots, f_d \rangle$*
- *$r_2 \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ or r_2 has a compatibility sequence $\langle g_1, \ldots, g_e \rangle$*
- *If $r_1, r_2 \notin \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, 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 $r_1 \notin \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, then for every f_d-rule α, terms in $M(\alpha) \backslash \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ 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[5] g_e-rule and $M(\alpha) = M_1(\alpha) \cup M_2(\alpha') \cup \{r_2[t_1^*], \ldots, r_2[t_n^*]\}$, where

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

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

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 $Exc_{\mathsf{dbl},+} = \{\alpha_1^+\}$ and $Pos_+(\alpha_1^+) = \{2\}$, $M(\alpha_1^+)$ 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 $Pos_{\mathsf{dbl},+}(\alpha_2^+) = Pos_+(\alpha_2^+) = \varnothing$, $M(\alpha_2^+)$ 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 $Pos_*(\alpha_1^*) = Pos_{+,(*,*)}(\alpha_1^*) = \varnothing$, $Pos_*(\alpha_2^*) = \{2\}$, $Pos_{+,(*,*)}(\alpha_2^*) = \{1\,2,\ 2\,2\}$, 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. 14 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 and generalizes all resulting proof obligations 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.[6]

[5] W.l.o.g, $r_1 \notin \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ unless $r_1, r_2 \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$. If $r_2 \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ then $M_2(...)$ is empty. Otherwise, for every f_d-rule α there is a *corresponding* g_e-rule α': $g_e(s^*, z^*) \to C'[g_e(t_1^*, z^*), ..., g_e(t_n^*, z^*)]$. We sometimes also write α instead of α'.

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

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.

Theorem 17 (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 [12] 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.

Checking whether an equation belongs to DEC can be done efficiently, since it mainly depends on the definitions of functions. Therefore, the required information can be pre-compiled. Moreover, for every equation in DEC, a failed induction proof attempt refutes the conjecture. So by restricting induction to equations from DEC, 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 Univ. Pr., 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. J. Giesl & D. Kapur. Deciding Inductive Validity of Equations. Technical Report AIB-2003-03, 2003. Available from `http://aib.informatik.rwth-aachen.de`
10. D. Kapur & H. Zhang. An Overview of Rewrite Rule Laboratory (*RRL*). *Journal of Computer and Mathematics with Applications*, 29:91–114, 1995.
11. D. Kapur & M. Subramaniam. New Uses of Linear Arithmetic in Automated Theorem Proving by Induction. *Journal of Automated Reasoning*, 16:39–78, 1996.
12. D. Kapur & M. Subramaniam. Extending Decision Procedures with Induction Schemes. *Proc. CADE-17*, LNAI 1831, pages 324-345, 2000.
13. M. Kaufmann, P. Manolios, & J S. Moore. *Computer-Aided Reasoning: An Approach*. Kluwer, 2000.
14. 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.
15. H. Zhang, D. Kapur, & M. S. Krishnamoorthy. A Mechanizable Induction Principle for Equational Specifications. *Proc. CADE-9*, LNCS 310, 1988.