Inductive Decidability Using Implicit Induction*

Stephan Falke and Deepak Kapur

Computer Science Department, University of New Mexico, Albuquerque, NM, USA
{spf, kapur}@cs.unm.edu

Abstract. Decision procedures are widely used in automated reasoning tools in order to reason about data structures. In applications, many conjectures fall outside the theory handled by a decision procedure. Often, reasoning about user-defined functions on those data structures is needed. For this, inductive reasoning has to be employed. In this work, classes of function definitions and conjectures are identified for which inductive validity can be automatically decided using implicit induction methods and decision procedures for an underlying theory. The class of equational conjectures considered in this paper significantly extends the results of Kapur & Subramaniam (CADE, 2000) [15], which were obtained using explicit induction schemes. Firstly, nonlinear conjectures can be decided automatically. Secondly, function definitions can use other defined functions in their definitions, thus allowing mutually recursive functions and decidable conjectures about them. Thirdly, conjectures can have general terms from the decidable theory on inductive positions. These contributions are crucial for successfully integrating inductive reasoning into decision procedures, thus enabling their use in push-button mode in applications including verification and program analysis.

1 Introduction

Inductive reasoning about recursively defined data structures and recursive functions defined on such data structures is often needed for verifying properties of computational descriptions, both in hardware and software. Decision procedures about commonly used data structures including numbers and data structures generated using free constructors (such as lists and trees) are being widely used in software and hardware verification. Push-button tools for program analysis and verification based on decision procedures have been explored. However, most tools based on decision procedures, including BLAST [9] and SLAM [3], are limited in their capabilities because of their inability to reason about recursively defined functions.

One of the major challenges is to integrate inductive reasoning with decision procedures and, in particular, to identify a subclass of inductive conjectures about recursively defined functions whose validity can be decided automatically. This line of research was initiated by Kapur, Giesl, and Subramaniam [15,7,8,12]. The aim of identifying inductive conjectures whose validity is decidable is achieved by imposing restrictions on the structure of function definitions,

* Partially supported by NSF grants ITR CCR-0113611 and CCR-0203051.

M. Hermann and A. Voronkov (Eds.): LPAR 2006, LNAI 4246, pp. 45–59, 2006.

as well as on the conjectures about these functions. Kapur *et al.* use the framework of explicit inductive reasoning based on the cover set method proposed in [20].

This paper uses the framework of implicit induction for automating inductive reasoning in order to extend the results given in [15]. Implicit induction methods, while less widely applicable, are considered to be more amenable to automation. A benefit of the implicit induction methods is that conjectures requiring mutual induction can be handled more easily than in the explicit induction framework.

In [15], the concept of theory-based functions is introduced. In the definition of a theory-based function, the only recursive calls permitted are to the same function again. The conjectures whose validity is shown decidable in [15] are equational conjectures $r_1 \approx r_2$, where r_2 is a term in the decidable theory and r_1 contains theory-based functions. The arguments to those theory-based functions are required to be distinct variables. In [7], Boolean combinations of those conjectures are considered. In [8], the class of conjectures is extended to linear equational conjectures containing theory-based functions on both sides.

In this paper, we substantially extend the results from [15] in three ways that are orthogonal to the extensions of [7,8]. Firstly, the permitted conjectures are generalized to include nonlinear conjectures and conjectures with general terms from the decidable theory on inductive positions, such as $\mathsf{gcd}(x, x) \approx x$ and $x \leq \mathsf{s}(x) \approx \mathsf{true}$, whose validity can be decided automatically. For handling such nonlinear conjectures, conditions on function definitions are identified which can easily be checked a priori using a decision procedure (Section 4). The second generalization is to allow the definition of a theory-based function to use theory-based functions other than the function being defined. A group of theory-based function symbols can thus be viewed as being defined jointly together. This extension allows for mutually recursive theory-based definitions (Section 5). As in [15], conjectures about these groups of function symbols can have nested function calls (Section 6). For each of these classes, a decision procedure based on implicit induction methods is given. The considered classes of conjectures are highly relevant in practice and can readily be handled using implicit induction methods. They are quite challenging for explicit inductive reasoning techniques.

Due to lack of space, almost all proofs are omitted. They may be found in the extended version of this paper [5], together with more details on the examples.

2 Background

We generally assume familiarity with the concepts of term rewriting [1]. We use many-sorted first-order logic where "$\approx$" is the only predicate symbol and "$\approx$" 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* $\mathcal{F}$ *and* $\mathcal{V}$ by $\mathcal{T}erms(\mathcal{F}, \mathcal{V})$ and the set of ground terms by $\mathcal{T}erms(\mathcal{F})$. We often write x^* to denote a tuple of (not necessarily pairwise distinct) variables, and denote by x_i the i^{th} element of this tuple. Analogously, s^* denotes a tuple of terms s_i.

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 (quantifier-free) theory $\mathcal{T}$ is defined to be the set of all quantifier-free formulas φ over $\mathcal{F}_\mathcal{T}$ such that $AX_\mathcal{T} \models \forall^*.\, \varphi$, where $\forall^*.\, \varphi$ is the universal closure of φ. In this case we also say that φ is *valid*. We often write $s \approx_\mathcal{T} t$ as a shorthand for $AX_\mathcal{T} \models \forall^*.\, s \approx t$ and $s \not\approx_\mathcal{T} t$ as a shorthand for $AX_\mathcal{T} \not\models \forall^*.\, s \approx t$.

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

$$\begin{array}{l}
\neg(\mathsf{c}(x_1,\ldots,x_n) \approx \mathsf{c}'(y_1,\ldots.y_m)) \text{ for all } \mathsf{c},\mathsf{c}' \in \mathcal{F}_{\mathcal{T}_C} \text{ where } \mathsf{c} \neq \mathsf{c}' \\
\mathsf{c}(x_1,\ldots,x_n) \approx \mathsf{c}(y_1,\ldots,y_n) \Longrightarrow x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n \text{ for all } \mathsf{c} \in \mathcal{F}_{\mathcal{T}_C} \\
\bigvee_{\mathsf{c}\in\mathcal{F}_{\mathcal{T}_C}} \exists y_1,\ldots,y_n.\, x \approx \mathsf{c}(y_1,\ldots,y_n) \\
\neg(\mathsf{c}_1(\ldots\mathsf{c}_2(\ldots\mathsf{c}_n(\ldots x\ldots)\ldots)\ldots) \approx x) \text{ for all sequences } \mathsf{c}_1\ldots\mathsf{c}_n \in \mathcal{F}^*_{\mathcal{T}_C}, n>0
\end{array}$$

Note that the last type of axioms usually results in infinitely many formulas. Here, "..." in the arguments of c_i stands for pairwise distinct variables.

We use the following definition for the equational sub-theory $\mathcal{T}_{PA}$ of *Presburger arithmetic* on natural numbers: $\mathcal{F}_{\mathcal{T}_{PA}} = \{\mathsf{0},\mathsf{1},+\}$ and $AX_{\mathcal{T}_{PA}}$ consists of the universal closures of the following formulas:

$$\begin{array}{ll}
(x+y)+z \approx x+(y+z) & \neg(\mathsf{1}+x \;\approx\; \mathsf{0}) \\
x+y \approx y+x & x+y \approx x+z \Longrightarrow y \approx z \\
\mathsf{0}+y \approx y & x \approx \mathsf{0} \;\vee\; \exists y.\, x \approx y+\mathsf{1}
\end{array}$$

We often write *flattened* terms since "+" is associative and commutative. For $t \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}_{PA}},\mathcal{V})$ with $\mathcal{V}(t) = \{x_1,\ldots,x_n\}$, there exist $a_i \in \mathbb{N}$ such that $t \approx_{\mathcal{T}_{PA}} a_0 + a_1\cdot x_1 + \ldots + a_n \cdot x_n$. Here, "$a \cdot x$" denotes the term $x+\ldots+x$ (a times) and "a_0" denotes $\mathsf{1}+\ldots+\mathsf{1}$ (a_0 times).

Instead of *validity*, we are usually interested in *inductive validity*. The formula φ is *inductively valid in a 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 from $\mathcal{T}erms(\mathcal{F}_\mathcal{T})$. 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 $r_1 \approx r_2$ over $\mathcal{F}_\mathcal{T}$ also implies its validity.

We assume that for all sets $U = \{s_1 \approx^?_\mathcal{T} t_1, \ldots, s_n \approx^?_\mathcal{T} t_n\}$ with $s_i, t_i \in \mathcal{T}erms(\mathcal{F}_\mathcal{T},\mathcal{V})$ for all $1 \leq i \leq n$, a finite minimal complete set $\mathcal{CU}_\mathcal{T}(U)$ of $\mathcal{T}$-unifiers is computable. If $U = \{s \approx^?_\mathcal{T} t\}$ we also write $\mathcal{CU}_\mathcal{T}(s,t)$.

We use *term rewrite systems* (TRSs) over a signature $\mathcal{F} \supseteq \mathcal{F}_\mathcal{T}$ as our specification language and require that all left sides of rules have the form $\mathsf{f}(s^*)$ for a tuple s^* of terms from $\mathcal{T}erms(\mathcal{F}_\mathcal{T},\mathcal{V})$ and $\mathsf{f} \notin \mathcal{F}_\mathcal{T}$. Let $\mathcal{F}_d = \mathcal{F} \setminus \mathcal{F}_\mathcal{T}$ denote the set of *defined symbols*. We use the concept of rewriting modulo a theory ($\rightarrow_{\mathcal{R}/\mathcal{T}}$), where $\rightarrow_{\mathcal{R}/\mathcal{T}}$ must be decidable. We restrict ourselves to terminating, confluent, and sufficiently complete TRSs $\mathcal{R}$, where $\mathcal{R}$ is *terminating* if $\rightarrow_{\mathcal{R}/\mathcal{T}}$ is well-founded, it is *confluent* if $\rightarrow_{\mathcal{R}/\mathcal{T}}$ is confluent, and it is *sufficiently complete* if for all (well-typed) ground terms $t \in \mathcal{T}erms(\mathcal{F})$ there exists a ground term $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T})$ such that $t \rightarrow^*_{\mathcal{R}/\mathcal{T}} q$. When regarding $\rightarrow^*_{\mathcal{R}/\mathcal{T}}$, we usually do not distinguish between terms that are equal w.r.t. $\approx_\mathcal{T}$.

A *reduction order* is a strict, well-founded order $\succ$ on $\mathcal{T}erms(\mathcal{F},\mathcal{V})$ that is closed under contexts ($C[s] \succ C[t]$ whenever $s \succ t$) and substitutions ($s\sigma \succ t\sigma$ whenever $s \succ t$). We say that $\succ$ is *compatible* with $\mathcal{T}$ if $\approx_{\mathcal{T}} \circ \succ \circ \approx_{\mathcal{T}} \subseteq \succ$, i.e., $\succ$ does not distinguish between terms that are equal w.r.t. $\approx_{\mathcal{T}}$.

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 $AX_{\mathcal{T}} \cup \{l \approx r \mid l \to r \in \mathcal{R}\}$. To ease readability, we write $AX_{\mathcal{T}} \cup \mathcal{R}$ instead of $AX_{\mathcal{T}} \cup \{l \approx r \mid l \to r \in \mathcal{R}\}$. This extension is *conservative*, i.e., it does not change inductive validity of equations over $\mathcal{F}_{\mathcal{T}}$ [8]. If $AX_{\mathcal{T}} \cup \mathcal{R} \models_{ind} r_1 \approx r_2$ we say that the equation $r_1 \approx r_2$ is an *inductive consequence* of $AX_{\mathcal{T}} \cup \mathcal{R}$.

3 Implicit Induction Methods and $\mathcal{T}$-Based Functions

Implicit induction is a proof method that was derived in [18] from the Knuth-Bendix completion procedure. Since its initial formulation, various improvements have been made to the basic method, see, e.g., [10,13,11,6,14,17,2,19,4].

In this paper, we follow the presentation in [19]. The results, however, are largely independent of this presentation and extend to other proposed methods like the ones in [14,2,4].

Here, as in [19], the implicit induction method is given by an inference system. The system $\mathcal{I}_{\mathcal{T}}$ shown in Figure 1 is parameterized by a TRS $\mathcal{R}$ and a reduction order $\succ$ which is compatible with $\mathcal{T}$ and orients $\mathcal{R}$. It operates on two sets of equations:

1. $\mathcal{E}$, containing the set of equations to be proven, and
2. $\mathcal{H}$, containing the equations (oriented as rewrite rules by $\succ$) which can be used as inductive hypotheses.

Here, $\mathcal{CP}(\mathcal{R}, l \to r)$ is the set of *critical pairs* of $\mathcal{R}$ on $l \to r$, i.e, $\mathcal{CP}(\mathcal{R}, l \to r) = \bigcup_{p \in \mathcal{FP}os(l)} \mathcal{CP}'(\mathcal{R}, l \to r, p)$, where $\mathcal{FP}os(l)$ denotes the non-variable positions in l and $\mathcal{CP}'(\mathcal{R}, l \to r, p) = \{r\sigma \approx l\sigma[r'\sigma]_p \mid l' \to r' \in \mathcal{R}, \sigma \in \mathcal{CU}_{\mathcal{T}}(l|_p, l')\}$. For this, the variables in $l' \to r'$ are suitably renamed to be disjoint from the variables in l. Notice that we use $\mathcal{T}$-unification in order to compute the critical pairs. We use $r_1 \stackrel{\cdot}{\approx} r_2$ to stand for either $r_1 \approx r_2$ or $r_2 \approx r_1$. The inference rules Theory_1 and Theory_2 are replaced by a decision procedure for $\mathcal{T}$ in practice.

The following theorem is obtained by extending [19, Proposition 18].

Theorem 1. *If there is a successful $\mathcal{I}_{\mathcal{T}}$-derivation $(\mathcal{E}_0, \emptyset) \vdash_{\mathcal{I}_{\mathcal{T}}} (\mathcal{E}_1, \mathcal{H}_1) \vdash_{\mathcal{I}_{\mathcal{T}}} \ldots \vdash_{\mathcal{I}_{\mathcal{T}}} (\emptyset, \mathcal{H}_n)$, then all equations in $\mathcal{E}_0$ are inductive consequences of $AX_{\mathcal{T}} \cup \mathcal{R}$. If there is a refuting $\mathcal{I}_{\mathcal{T}}$-derivation $(\mathcal{E}_0, \emptyset) \vdash_{\mathcal{I}_{\mathcal{T}}} (\mathcal{E}_1, \mathcal{H}_1) \vdash_{\mathcal{I}_{\mathcal{T}}} \ldots \vdash_{\mathcal{I}_{\mathcal{T}}} \bot$, then some equation in $\mathcal{E}_0$ is not an inductive consequence of $AX_{\mathcal{T}} \cup R$.*

In [15], the concept of a *$\mathcal{T}$-based* function is introduced: in the rewrite rules defining a function f, all arguments to f are terms from $\mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$, and the right side becomes a term in $\mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$ after subterms of the form $\mathsf{f}(t^*)$, if any, are abstracted using new variables.

$$\textsf{Expand}\ \frac{\mathcal{E} \cup \{r_1 \stackrel{\cdot}{\approx} r_2\}, \mathcal{H}}{\mathcal{E} \cup \mathcal{E}', \mathcal{H} \cup \{r_1 \to r_2\}} \quad \text{if } r_1 \succ r_2 \text{ and } \mathcal{E}' = \mathcal{CP}(\mathcal{R}, r_1 \to r_2)$$

$$\textsf{Simplify}\ \frac{\mathcal{E} \cup \{r_1 \stackrel{\cdot}{\approx} r_2\}, \mathcal{H}}{\mathcal{E} \cup \{r_1' \approx r_2\}, \mathcal{H}} \quad \text{if } r_1 \to_{\mathcal{R} \cup \mathcal{H}/\mathcal{T}} r_1'$$

$$\textsf{Theory}_1\ \frac{\mathcal{E} \cup \{r_1 \approx r_2\}, \mathcal{H}}{\mathcal{E}, \mathcal{H}} \quad \text{if } r_1 \approx_{\mathcal{T}} r_2 \qquad \textsf{Theory}_2\ \frac{\mathcal{E} \cup \{r_1 \approx r_2\}, \mathcal{H}}{\bot} \quad \text{if } r_1 \not\approx_{\mathcal{T}} r_2$$

Fig. 1. The inference system $\mathcal{I}_{\mathcal{T}}$

Definition 2 ($\mathcal{T}$-based Functions [15]). *A function* $\mathsf{f} \in \mathcal{F}_d$ *is* $\mathcal{T}$-based *if all rules* $l \to r \in \mathcal{R}$ *with* $l(\Lambda) = \mathsf{f}$ *have the form* $\mathsf{f}(s^*) \to C[\mathsf{f}(t_1^*), \ldots, \mathsf{f}(t_n^*)]$, *where* $s^*, t_1^*, \ldots, t_n^* \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$, *and* C *is a context over* $\mathcal{F}_{\mathcal{T}}$.

The *inductive positions* of a function f are those positions such that subterms on those positions change by applying the rules defining f.

Definition 3 (Inductive Positions). *For a* $\mathcal{T}$*-based* f, *position* i *with* $1 \leq i \leq \mathsf{arity}(\mathsf{f})$ *is* non-inductive *if for all rules* $\mathsf{f}(s_1, \ldots, s_m) \to C[\mathsf{f}(t_{1,1}, \ldots, t_{1,m}), \ldots, \mathsf{f}(t_{n,1}, \ldots, t_{n,m})]$ *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 all* $1 \leq k \leq n$. *Otherwise, the position is* inductive.

We generally assume that the first n positions of any function f are inductive, while the remaining positions are non-inductive, for some $0 \leq n \leq \mathsf{arity}(\mathsf{f})$.

Example 4. Let $\mathcal{R}$ be the TRS defining "$-$" on natural numbers over $\mathcal{T}_C$.

$$1 : x - \mathsf{0} \to x \qquad 2 : \mathsf{0} - \mathsf{s}(y) \to \mathsf{0} \qquad 3 : \mathsf{s}(x) - \mathsf{s}(y) \to x - y$$

Then both position 1 and position 2 of "$-$" are inductive positions. ◊

4 Nonlinear Simple Conjectures

In [15], a class of conjectures about theory-based functions of the form $\mathsf{f}(x^*, s^*) \approx r$, where all variables in x^* are pairwise distinct and do not occur in s^*, is defined. This rules out nonlinear conjectures like $x' - x' \approx \mathsf{0}$. The restrictions were imposed to ensure that the inductive hypothesis is applicable. Below, we show how the restriction on linearity in the left side can be relaxed, thus substantially expanding the class of conjectures which can be decided automatically.

Example 5. Continuing Example 4, we want to prove $x' - x' \approx \mathsf{0}$. Indeed,

$$\begin{aligned}(\{x' - x' \approx \mathsf{0}\}, \emptyset) &\vdash_{\textsf{Expand}} (\{\mathsf{0} \approx \mathsf{0}, \mathsf{0} \approx x - x\}, \{x' - x' \to \mathsf{0}\}) \\ &\vdash_{\textsf{Simplify}} (\{\mathsf{0} \approx \mathsf{0}\}, \{x' - x' \to \mathsf{0}\}) \\ &\vdash_{\textsf{Theory}_1} (\emptyset, \{x' - x' \to \mathsf{0}\})\end{aligned}$$

is a successful derivation. ◊

The reason this derivation is successful is that whenever the terms on the inductive positions in the left side of a rule have a relationship (in this case, equality), the terms on those positions of the recursive calls in the right side have the same relationship, which is needed to apply the inductive hypothesis. This observation is formalized using the set $\mathit{Imp}\mathcal{E}q(\mathsf{f})$.

Definition 6 ($\mathit{Imp}\mathcal{E}q(\mathsf{f})$). *Let* f *be* $\mathcal{T}$*-based and defined by the rules* $\mathsf{f}(s_i^*) \to C_i[\mathsf{f}(t_{i,1}^*), \ldots, \mathsf{f}(t_{i,n_i}^*)]$ *for* $1 \leq i \leq n$*. Then we define* $\langle l_1, l_2, \mathcal{C}\rangle \in \mathit{Imp}\mathcal{E}q(\mathsf{f})$ *iff* $\mathcal{C} = \{\langle k_{j,1}, k_{j,2}\rangle \mid 1 \leq j \leq m\}$ *for some* m *such that* $1 \leq l_1 < l_2 \leq \mathsf{arity}(\mathsf{f})$*,* $1 \leq k_{j,1} < k_{j,2} \leq \mathsf{arity}(\mathsf{f})$ *for all* $1 \leq j \leq m$*,*

$$\bigwedge_{j=1}^{m} s_{i,k_{j,1}} \approx s_{i,k_{j,2}} \Longrightarrow \bigwedge_{j=1}^{n_i} t_{i,j,l_1} \approx t_{i,j,l_2}$$

is $\mathcal{T}$*-valid for all* $1 \leq i \leq n$*, and there is no* $\mathcal{C}' \subsetneq \mathcal{C}$ *with this property.*

Hence, if a term of the form $\mathsf{f}(s_i^*)\sigma$ is simplified using the rule $\mathsf{f}(s_i^*) \to C_i[\mathsf{f}(t_{i,1}^*), \ldots, \mathsf{f}(t_{i,n_i}^*)]$, then $t_{i,k,l_1}\sigma \approx_{\mathcal{T}} t_{i,k,l_2}\sigma$ for all $1 \leq k \leq n_i$ if $s_{i,k_{j,1}}\sigma \approx_{\mathcal{T}} s_{i,k_{j,2}}\sigma$ for all $1 \leq j \leq m$.

Example 7. Continuing Example 5, we get $\mathit{Imp}\mathcal{E}q(-) = \{\langle 1, 2, \{\langle 1, 2\rangle\}\rangle\}$. For rule 1, $x \approx \mathsf{0} \Longrightarrow \mathtt{true}$ is obviously $\mathcal{T}_C$-valid. Also, $\mathsf{0} \approx \mathsf{s}(y) \Longrightarrow \mathtt{true}$ obtained from rule 2 is $\mathcal{T}_C$-valid. For rule 3, $\mathsf{s}(x) \approx \mathsf{s}(y) \Longrightarrow x \approx y$ is $\mathcal{T}_C$-valid as well. $\Diamond$

Clearly, the set $\mathit{Imp}\mathcal{E}q(\mathsf{f})$ can be computed at compile-time from the rules defining f with the help of a decision procedure for $\mathcal{T}$.

Below, we extend the definition of simple conjectures in [15] to consider nonlinear conjectures by relaxing the requirement that variables on inductive positions need to be pairwise distinct.

Definition 8 (Simple Conjectures). *A* simple conjecture *is a conjecture of the form* $\mathsf{f}(x^*, s^*) \approx r$ *such that*[1] $\mathsf{f}(x^*, s^*) \succ r$*, the function* f *is* $\mathcal{T}$*-based,* $s^*, r \in \mathcal{T}\mathit{erms}(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$*, the* x_i *are on* f*'s inductive positions and do not appear in the* s_j*, and if* $x_{l_1} = x_{l_2}$ *then there exists* $\langle l_1, l_2, \mathcal{C}\rangle \in \mathit{Imp}\mathcal{E}q(\mathsf{f})$ *such that* $x_{k_1} = x_{k_2}$ *for all* $\langle k_1, k_2\rangle \in \mathcal{C}$*.*

The following theorem gives a decision procedure based on the implicit induction framework for the class of simple conjectures which includes nonlinear as well as linear conjectures.

Theorem 9. *The inductive validity of a simple conjecture is decidable using the strategy*[2] $\mathsf{Expand} \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_1 \cup \mathsf{Theory}_2)^*$.

Proof. Let $\mathsf{f}(x^*, s^*) \approx r$ be a simple conjecture, and let

[1] Here, $\succ$ is the same reduction order used to orient $\mathcal{R}$. In practice this means that $\mathcal{R} \cup \{\mathsf{f}(x^*, s^*) \to r\}$ is terminating.

[2] Here, $\cdot^*$ means that the inference rule is applied exhaustively.

$$\begin{aligned}\mathcal{R}' = \{\ & \mathsf{f}(s_1^*, y^*) \to C_1[\mathsf{f}(t_{1,1}^*, y^*), \ldots, \mathsf{f}(t_{1,n_1}^*, y^*)],\\ & \ldots,\\ & \mathsf{f}(s_m^*, y^*) \to C_m[\mathsf{f}(t_{m,1}^*, y^*), \ldots, \mathsf{f}(t_{m,n_m}^*, y^*)]\ \}\end{aligned}$$

be the definition of the $\mathcal{T}$-based function f.

Applying Expand to $(\{\mathsf{f}(x^*, s^*) \approx r\}, \emptyset)$, we obtain the state

$$\begin{aligned}\mathcal{E} &= \{\ C_{i_1}[\mathsf{f}(t_{i_1,1}^*, y^*), \ldots, \mathsf{f}(t_{i_1,n_{i_1}}^*, y^*)]\sigma_{i_1} \approx r\sigma_{i_1},\\ &\quad \ldots,\\ &\quad C_{i_l}[\mathsf{f}(t_{i_l,1}^*, y^*), \ldots, \mathsf{f}(t_{i_l,n_{i_l}}^*, y^*)]\sigma_{i_l} \approx r\sigma_{i_l} \quad \},\\ \mathcal{H} &= \{\ \mathsf{f}(x^*, s^*) \to r \quad \}\end{aligned}$$

for some $i_1, \ldots, i_l \in \{1, \ldots, m\}$, where $\sigma_{i_j} \in \mathcal{CU}_{\mathcal{T}}(\{x^* \approx_{\mathcal{T}}^? s_{i_j}^*, y^* \approx_{\mathcal{T}}^? s^*\})$. The i_j are such that the arguments of $\mathsf{f}(x^*, s^*)$ and $\mathsf{f}(s_{i_j}^*, y^*)$ are $\mathcal{T}$-unifiable. To ease readability, we assume that every complete set of unifiers has cardinality 1.

If $n_{i_j} = 0$, then $C_{i_j}[\mathsf{f}(t_{i_j,1}^*, y^*), \ldots, \mathsf{f}(t_{i_j,n_{i_j}}^*, y^*)]\sigma_{i_j} = C_{i_j}\sigma_{i_j} \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$. Hence, either Theory_1 or Theory_2 applies to $C_{i_j}\sigma_{i_j} \approx r\sigma_{i_j}$.

If $n_{i_j} > 0$, then

$$\begin{aligned}&C_{i_j}[\mathsf{f}(t_{i_j,1}^*, y^*), \ldots, \mathsf{f}(t_{i_j,n_{i_j}}^*, y^*)]\sigma_{i_j}\\ &= C_{i_j}\sigma_{i_j}[\mathsf{f}(t_{i_j,1}^*\sigma_{i_j}, s^*), \ldots, \mathsf{f}(t_{i_j,n_{i_j}}^*\sigma_{i_j}, s^*)].\end{aligned}$$

Now, if $x_{l_1} = x_{l_2}$, then there exists $\langle l_1, l_2, \mathcal{C}\rangle \in \mathcal{I}mp\mathcal{E}q(\mathsf{f})$ such that $x_{k_1} = x_{k_2}$ for all $\langle k_1, k_2\rangle \in \mathcal{C}$. But since the arguments of $\mathsf{f}(x^*, s^*)$ and $\mathsf{f}(s_{i_j}^*, y^*)$ are $\mathcal{T}$-unifiable by σ_{i_j}, this means $s_{i_j,k_1}\sigma_{i_j} \approx_{\mathcal{T}} s_{i_j,k_2}\sigma_{i_j}$ for all $\langle k_1, k_2\rangle \in \mathcal{C}$. Now, the definition of $\mathcal{I}mp\mathcal{E}q(\mathsf{f})$ implies $t_{i_j,k,l_1}\sigma_{i_j} \approx_{\mathcal{T}} t_{i_j,k,l_2}\sigma_{i_j}$ for all $1 \le k \le n_{i_j}$.

Hence, $\mathsf{Simplify}$ applies n_{i_j} times to $C_{i_j}\sigma_{i_j}[\mathsf{f}(t_{i_j,1}^*\sigma_{i_j}, s^*), \ldots, \mathsf{f}(t_{i_j,n_{i_j}}^*\sigma_{i_j}, s^*)]$ using the rule $\mathsf{f}(x^*, s^*) \to r \in \mathcal{H}$, to get $C_{i_j}\sigma_{i_j}[r\tau_{i_j,1}, \ldots, r\tau_{i_j,n_{i_j}}] \approx r\sigma_{i_j}$, where $\tau_{i_j,k} = \{x^* \mapsto t_{i_j,k}^*\sigma_{i_j}\}$. Since both sides are in $\mathcal{T}erms(\mathcal{F}_{\mathcal{T}}, \mathcal{V})$, either Theory_1 or Theory_2 applies. □

With this theorem many conjectures can be handled which did not fall in the class of simple conjectures in [15].

Example 10. Consider this TRS over $\mathcal{T}_C$.

$$\begin{array}{lll}\mathsf{max}(0, y) \to y & \mathsf{max}(\mathsf{s}(x), 0) \to \mathsf{s}(x) & \mathsf{max}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(\mathsf{max}(x, y))\\ \mathsf{min}(0, y) \to 0 & \mathsf{min}(\mathsf{s}(x), 0) \to 0 & \mathsf{min}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(\mathsf{min}(x, y))\\ x < 0 \to \mathsf{false} & 0 < \mathsf{s}(y) \to \mathsf{true} & \mathsf{s}(x) < \mathsf{s}(y) \to x < y\\ 0 \le y \to \mathsf{true} & \mathsf{s}(x) \le 0 \to \mathsf{false} & \mathsf{s}(x) \le \mathsf{s}(y) \to x \le y\end{array}$$

Then, the following conjectures can be decided using Theorem 9:

$$\begin{array}{ll}\mathsf{min}(x, x) \approx x & \mathsf{max}(x, x) \approx x\\ x < x \approx \mathsf{false} & x \le x \approx \mathsf{true}\\ x < x \approx \mathsf{true} & x \le x \approx \mathsf{false}\end{array}$$

For this, we notice that $\mathcal{I}mp\mathcal{E}q(\mathsf{f}) = \{\langle 1, 2, \{\langle 1, 2\rangle\}\rangle\}$ for $\mathsf{f} \in \{\mathsf{min}, \mathsf{max}, <, \le\}$. ◊

4.1 Relaxing $\mathcal{I}mp\mathcal{E}q$

In some cases, the conditions of Definition 8 can be relaxed. Firstly, there can be conjectures for which the inductive hypotheses need not be applied if the recursive calls already rewrite to terms in the decidable theory using $\mathcal{R}$.

Example 11. Consider the TRS defining gcd over $\mathcal{T}_{PA}$.

$$\begin{array}{ll} 1: & \mathsf{gcd}(x,0) \to x \\ 2: & \mathsf{gcd}(0,y+1) \to y+1 \end{array} \qquad \begin{array}{ll} 3: & \mathsf{gcd}(x+y+1,y+1) \to \mathsf{gcd}(x,y+1) \\ 4: & \mathsf{gcd}(x+1,x+y+1) \to \mathsf{gcd}(x+1,y) \end{array}$$

Assume the conjecture $\mathsf{gcd}(x',x') \approx x'$ is to be proven. Theorem 9 cannot be used, since $\langle 1,2,\{1,2\}\rangle \notin \mathcal{I}mp\mathcal{E}q(\mathsf{gcd})$ because $x+y+1 \approx y+1 \Longrightarrow x \approx y+1$, which is obtained from rule 3, is not $\mathcal{T}_{PA}$-valid.

However, there is a successful derivation.

$$\begin{array}{ll} & (\{\mathsf{gcd}(x',x') \approx x'\},\emptyset) \\ \vdash_{\mathsf{Expand}} & (\{0 \approx 0, y+1 \approx \mathsf{gcd}(0,y+1), x+1 \approx \mathsf{gcd}(x+1,0)\}, \{\mathsf{gcd}(x',x') \to x'\}) \\ \vdash_{\mathsf{Simplify}} & (\{0 \approx 0, y+1 \approx y+1, x+1 \approx \mathsf{gcd}(x+1,0)\}, \{\mathsf{gcd}(x',x') \to x'\}) \\ \vdash_{\mathsf{Simplify}} & (\{0 \approx 0, y+1 \approx y+1, x+1 \approx x+1\}, \{\mathsf{gcd}(x',x') \to x\}) \\ \vdash^{3}_{\mathsf{Theory}_1} & (\emptyset, \{\mathsf{gcd}(x',x') \to x'\}) \end{array}$$

The key observation is that the recursive calls that are generated by Expand simplify to terms in $\mathcal{T}erms(\mathcal{F}_{\mathcal{T}_{PA}},\mathcal{V})$ using just rewrite rules in $\mathcal{R}$, without using the inductive hypothesis. ◊

Secondly, general terms from a decidable theory can be allowed on inductive positions in a conjecture if it can be ensured that the hypotheses will be applicable if they are needed. Notice that applicability of a hypothesis means that it $\mathcal{T}$-matches the recursive call generated by Expand.

Example 12. Continuing Example 10, we attempt to prove the conjecture $x' < \mathsf{s}(x') \approx \mathsf{true}$, which is nonlinear and has the term $\mathsf{s}(x')$ on an inductive position. The proof attempt is as follows:

$$\begin{array}{ll} (\{x' < \mathsf{s}(x') \approx \mathsf{true}\},\emptyset) \vdash_{\mathsf{Expand}} & (\{\mathsf{true} \approx \mathsf{true}, \mathsf{true} \approx x < \mathsf{s}(x)\}, \{x' < \mathsf{s}(x') \to \mathsf{true}\}) \\ \vdash_{\mathsf{Simplify}} & (\{\mathsf{true} \approx \mathsf{true}\}, \{x' < \mathsf{s}(x') \to \mathsf{true}\}) \\ \vdash_{\mathsf{Theory}_1} & (\emptyset, \{x' < \mathsf{s}(x') \to \mathsf{true}\}). \end{array}$$

The key observation is that the recursive call generated by Expand has the "right" form for the inductive hypothesis to apply, i.e., $(x<y)\sigma$ has the form $z < \mathsf{s}(z)$ where σ is the substitution generated by Expand from the third rule defining "<", i.e, $\sigma = \{y \mapsto \mathsf{s}(x), x' \mapsto \mathsf{s}(x)\}$. ◊

For formalizing these observations, we define the set $\mathcal{TP}at(\mathsf{f})$. In the following, we identify two tuples p^*, q^* of the same length containing terms from $\mathcal{T}erms(\mathcal{F}_{\mathcal{T}},\mathcal{V})$ if q^* can be obtained from p^* by means of a variable renaming.

Definition 13 ($\mathcal{TP}at(\mathsf{f})$)**.** *Let* f *be* $\mathcal{T}$*-based and defined by the rules* $\mathsf{f}(s_i^*) \to C_i[\mathsf{f}(t_{i,1}^*), \ldots, \mathsf{f}(t_{i,n_i}^*)]$ *for* $1 \le i \le n$*. Then we define* $p^* \in \mathcal{TP}at(\mathsf{f})$ *for* $p^* \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ *iff for all* $1 \le i \le n$*, all* $\sigma \in \mathcal{CU}_\mathcal{T}(s_i^* \approx^?_\mathcal{T} p^*)$[3] *and all* $1 \le k \le n_i$ *either*

1. p^* $\mathcal{T}$*-matches* $t_{i,k}^*\sigma$*, or*
2. $\mathsf{f}(t_{i,k}^*)\sigma \to^*_{\mathcal{R}/\mathcal{T}} q$ *for some* $q \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$*.*

Condition (1.) ensures that the inductive hypothesis is applicable, as demonstrated in Example 12. Condition (2.) is for cases like Example 11 where the inductive hypothesis is not needed. Also, notice that condition (1.) subsumes $\mathcal{I}mp\mathcal{E}q(\mathsf{f})$.

The set $\mathcal{TP}at(\mathsf{f})$ is in general infinite. However, it does not have to be computed at compile-time. Instead, it can be generated *lazily* as needed and cached for later reuse.

Definition 14 (Generalized Simple Conjectures). *A* generalized simple conjecture *is a conjecture of the form* $\mathsf{f}(p^*) \approx r$ *such that* $\mathsf{f}(p^*) \succ r$*, the function* f *is* $\mathcal{T}$*-based,* $r \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$*, and* $p^* \in \mathcal{TP}at(\mathsf{f})$*.*

Example 15. Continuing Example 11, the conjecture $\mathsf{gcd}(x', x') \approx x'$ is generalized simple since $(x', x') \in \mathcal{TP}at(\mathsf{gcd})$. Indeed, for rule 3, $\mathcal{CU}_{\mathcal{T}_{PA}}(\{x' \approx^?_{\mathcal{T}_{PA}} x + y + 1, x' \approx^?_{\mathcal{T}_{PA}} y + 1\}) = \{\sigma\}$ where $\sigma = \{x \mapsto \mathsf{0}, x' \mapsto y + 1\}$, and for this σ we have $\mathsf{gcd}(x, y + 1)\sigma = \mathsf{gcd}(\mathsf{0}, y + 1) \to_{\mathcal{R}/\mathcal{T}_{PA}} y + 1 \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}_{PA}}, \mathcal{V})$, i.e., (2.) in Definition 13 applies. For rule 4, we get $\sigma = \{y \mapsto \mathsf{0}, x' \mapsto x + 1\}$ and $\mathsf{gcd}(x + 1, y)\sigma = \mathsf{gcd}(x + 1, \mathsf{0}) \to_{\mathcal{R}/\mathcal{T}} x + 1 \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}_{PA}}, \mathcal{V})$. ◊

The following theorem gives a decision procedure for generalized simple conjectures, a substantially expanded class compared to the class of simple conjectures in [15], which allows nonlinear conjectures as well as conjectures in which general terms from a decidable theory can appear on inductive positions.

Theorem 16. *The inductive validity of a generalized simple conjecture is decidable using the strategy* $\mathsf{Expand} \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_1 \cup \mathsf{Theory}_2)^*$*.*

Example 17. Continuing Example 12, the conjecture $x' < \mathsf{s}(x') \approx \mathsf{true}$ is generalized simple since $(x', \mathsf{s}(x')) \in \mathcal{TP}at(<)$. Indeed, for $\mathsf{s}(x) < \mathsf{s}(y) \to x < y$, we get $\mathcal{CU}_{\mathcal{T}_C}(\{\mathsf{s}(x) \approx^?_{\mathcal{T}_C} x', \mathsf{s}(y) \approx^?_{\mathcal{T}_C} \mathsf{s}(x')\}) = \{\sigma\}$ where $\sigma = \{x' \mapsto \mathsf{s}(x), y \mapsto \mathsf{s}(x)\}$, and $(x < y)\sigma = x < \mathsf{s}(x)$, where $(x, \mathsf{s}(x))$ is $\mathcal{T}_C$-matched by $(x', \mathsf{s}(x'))$. ◊

Example 18. Continuing Example 11, we consider the conjecture $\mathsf{gcd}(2 \cdot x', 2) \approx 2$ about gcd. We show that $(2 \cdot x', 2) \in \mathcal{TP}at(\mathsf{gcd})$. For rules 1 and 2, there are no recursive calls on the right sides. For rule 3, $\mathcal{CU}_{\mathcal{T}_{PA}}(\{2 \cdot x' \approx^?_{\mathcal{T}_{PA}} x + y + 1, 2 \approx^?_{\mathcal{T}_{PA}} y + 1\}) = \{\sigma\}$, where $\sigma = \{x' \mapsto z + 1, y \mapsto 1, x \mapsto 2 \cdot z\}$ for a fresh variable z. Now, $\mathsf{gcd}(x, y + 1)\sigma = \mathsf{gcd}(2 \cdot z, 2)$, where $(2 \cdot z, 2)$ is $\mathcal{T}_{PA}$-matched by $(2 \cdot x', 2)$. Rule 4 yields $\mathcal{CU}_{\mathcal{T}_{PA}}(\{2 \cdot x' \approx^?_{\mathcal{T}_{PA}} x + 1, 2 \approx^?_{\mathcal{T}_{PA}} x + y + 1\}) = \{\sigma\}$, with $\sigma = \{x' \mapsto 1, x \mapsto 1, y \mapsto \mathsf{0}\}$. Now, $\mathsf{gcd}(x + 1, y)\sigma = \mathsf{gcd}(2, \mathsf{0}) \to_{\mathcal{R}/\mathcal{T}} 2 \in \mathcal{T}erms(\mathcal{F}_{\mathcal{T}_{PA}}, \mathcal{V})$. Thus, the conjecture $\mathsf{gcd}(2 \cdot x', 2) \approx 2$ is generalized simple. ◊

[3] The variables in p^* are suitably renamed to be disjoint from the variables in s_i^*.

5 Jointly $\mathcal{T}$-Based Functions

The class of $\mathcal{T}$-based functions is quite restrictive in the sense that a $\mathcal{T}$-based function f can only make recursive calls to f, but not to any other function g. This restriction is imposed to ensure that the strategy $\mathsf{Expand} \cdot \mathsf{Simplify}^*$ leads to conjectures in $\mathcal{T}$. Below, we generalize the definition of a $\mathcal{T}$-based function by allowing a finite number of $\mathcal{T}$-based functions to be defined together as a group, where a function in the group may call other functions in the group in its definition. This generalization allows for mutually recursive functions and classes of conjectures about them that can be decided automatically.

Example 19. We define $\mathsf{maxlist}$ using the function max from Example 10.

$$\mathsf{maxlist}(x, \mathsf{nil}) \to \mathsf{nil} \qquad \mathsf{maxlist}(\mathsf{nil}, \mathsf{cons}(x', y')) \to \mathsf{nil}$$
$$\mathsf{maxlist}(\mathsf{cons}(x, y), \mathsf{cons}(x', y')) \to \mathsf{cons}(\mathsf{max}(x, x'), \mathsf{maxlist}(y, y'))$$

While max is $\mathcal{T}_C$-based, $\mathsf{maxlist}$ is not $\mathcal{T}_C$-based since it calls max. Hence, the conjecture $\mathsf{maxlist}(x'', x'') \approx x''$ is not simple. Indeed, an attempt to prove this conjecture gets stuck since one ends up with $\mathsf{cons}(x, y) \approx \mathsf{cons}(\mathsf{max}(x, x), y)$, which cannot be simplified without knowing $\mathsf{max}(x, x) \approx x$ in conjunction with the original conjecture. ◊

Below, we relax the restriction on $\mathcal{T}$-based functions by introducing the notion of a *jointly $\mathcal{T}$-based* set of functions. A set F of functions is jointly $\mathcal{T}$-based if the rules defining F make recursive calls only to functions in F again. It is now possible for a recursive definition to use other functions in the group as well.

Definition 20 (Jointly $\mathcal{T}$-based Functions). *The set $\mathsf{F} = \{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ of functions $\mathsf{f}_i \in \mathcal{F}_d$ is* jointly $\mathcal{T}$-based *if all rules $l \to r \in \mathcal{R}$ with $l(\Lambda) \in \mathsf{F}$ have the form $\mathsf{f}_i(s^*) \to C[\mathsf{f}_{i_1}(t_1^*), \ldots, \mathsf{f}_{i_n}(t_n^*)]$, where $s^*, t_j^* \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, $\mathsf{f}_{i_j} \in \mathsf{F}$ for all $1 \leq j \leq n$, and C is a context over $\mathcal{F}_\mathcal{T}$.*

In particular, this definition allows for, but is not limited to, mutually recursive functions. Clearly, f is $\mathcal{T}$-based iff $\{\mathsf{f}\}$ is jointly $\mathcal{T}$-based, hence Definition 20 subsumes Definition 2.

To handle nonlinear conjectures for jointly $\mathcal{T}$-based functions, $\mathcal{I}mp\mathcal{E}q(\mathsf{f})$ does not suffice in order to guarantee that the inductive hypotheses are applicable. Instead, the whole set F of jointly $\mathcal{T}$-based functions has to be taken into consideration. The set $\mathcal{I}mp\mathcal{E}q(\mathsf{F})$ generalizes the set $\mathcal{I}mp\mathcal{E}q(\mathsf{f})$ by carrying the function symbols along with the positions $l_1, l_2, k_{j,1}, k_{j,2}$ from Definition 6. A formal definition of the set $\mathcal{I}mp\mathcal{E}q(\mathsf{F})$ can be found in [5, Definition 17]. If $\mathsf{F} = \{\mathsf{f}\}$, we identify $\mathcal{I}mp\mathcal{E}q(\mathsf{F})$ and $\mathcal{I}mp\mathcal{E}q(\mathsf{f})$. The set $\mathcal{I}mp\mathcal{E}q(\mathsf{F})$ can be computed from $\mathcal{R}$ with the help of a decision procedure for $\mathcal{T}$. See [5, Appendix A] for details. $\mathcal{TP}at(\mathsf{f})$ can similarly be extended to $\mathcal{TP}at(\mathsf{F})$.

Example 21. Considering the jointly $\mathcal{T}$-based set $\mathsf{F} = \{\mathsf{max}, \mathsf{maxlist}\}$ from Example 19, we get $\mathcal{I}mp\mathcal{E}q(\mathsf{F}) = \{\langle \mathsf{max}, 1, 2, \{\langle \mathsf{max}, 1, 2\rangle, \langle \mathsf{maxlist}, 1, 2\rangle\}\rangle, \langle \mathsf{maxlist}, 1, 2, \{\langle \mathsf{maxlist}, 1, 2\rangle\}\rangle\}$ since all of $\mathsf{s}(x) \approx \mathsf{s}(y) \Longrightarrow x \approx y$, $\mathsf{cons}(x, y) \approx \mathsf{cons}(x', y') \Longrightarrow x \approx x'$ (for max), and $\mathsf{cons}(x, y) \approx \mathsf{cons}(x', y') \Longrightarrow y \approx y'$ (for $\mathsf{maxlist}$) are $\mathcal{T}_C$-valid. ◊

Now, a *conjunctive simple conjecture* has to make a conjecture about each function in a set F of jointly $\mathcal{T}$-based functions.

Definition 22 (Conjunctive Simple Conjectures). *A* conjunctive simple conjecture *is a conjecture of the form* $\mathsf{f}_1(x_1^*) \approx r_1 \wedge \ldots \wedge \mathsf{f}_n(x_n^*) \approx r_n$ *such that* $\mathsf{f}_i(x_i^*) \succ r_i$ *for all* $1 \leq i \leq n$*, the set* $\mathsf{F} = \{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ *is jointly* $\mathcal{T}$*-based,* $r_i \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$ *for all* $1 \leq i \leq n$*, and if* $x_{i,l_1} = x_{i,l_2}$ *then exists* $\langle \mathsf{f}_i, l_1, l_2, \mathcal{C}\rangle \in \mathcal{I}mp\mathcal{E}q(\mathsf{F})$ *such that* $x_{j,k_1} = x_{j,k_2}$ *for all* $\langle \mathsf{f}_j, k_1, k_2\rangle \in \mathcal{C}$*.*

Example 23. Continuing Example 21, $\mathsf{maxlist}(x'', x'') \approx x'' \wedge \mathsf{max}(y'', y'') \approx y''$ is a conjunctive simple conjectures. ◊

The following theorem generalizes Theorem 9 and gives a decision procedure for conjunctive simple conjectures. An analogous theorem can be obtained for *conjunctive generalized simple conjectures* which use $\mathcal{TP}at$ instead of $\mathcal{I}mp\mathcal{E}q$ [5].

Theorem 24. *The inductive validity of a conjunctive simple conjecture is decidable using the strategy* $\mathsf{Expand}^* \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_1 \cup \mathsf{Theory}_2)^*$*, where* Expand *is applied once to each equation of the conjecture.*

Example 25. Consider the following TRS over $\mathcal{T}_C$, where mix takes two lists l_1, l_2 and constructs a new list by taking elements on odd numbered positions from l_1 and elements on even numbered positions from l_2.

$$\begin{array}{ll} \mathsf{mix}(x, \mathsf{nil}) \to \mathsf{nil} & \mathsf{mix}(\mathsf{nil}, \mathsf{cons}(x', y')) \to \mathsf{nil} \\ \mathsf{mix}(\mathsf{cons}(x, y), \mathsf{cons}(x', y')) \to \mathsf{cons}(x, \mathsf{mix}'(y, y')) & \\ \mathsf{mix}'(x, \mathsf{nil}) \to \mathsf{nil} & \mathsf{mix}'(\mathsf{nil}, \mathsf{cons}(x', y')) \to \mathsf{nil} \\ \mathsf{mix}'(\mathsf{cons}(x, y), \mathsf{cons}(x', y')) \to \mathsf{cons}(x', \mathsf{mix}(y, y')) & \end{array}$$

$\langle \mathsf{mix}, 1, 2, \{\langle \mathsf{mix}', 1, 2\rangle\}\rangle$ and $\langle \mathsf{mix}', 1, 2, \{\langle \mathsf{mix}, 1, 2\rangle\}\rangle$ are in $\mathcal{I}mp\mathcal{E}q(\{\mathsf{mix}, \mathsf{mix}'\})$. Thus, $\mathsf{mix}(x'', x'') \approx x'' \wedge \mathsf{mix}'(y'', y'') \approx y''$ is a conjunctive simple conjecture.◊

6 Nonlinear Complex Conjectures

So far, we have considered conjectures in which only a single defined function symbol appears in the left side of each equation within a conjecture. Similar to [15,8], conjectures in which the left side has nested $\mathcal{T}$-based functions can also be decided automatically provided their definitions are *compatible*. Furthermore, the results from the previous two sections extend to such complex conjectures under the same conditions as in [15]. Below, we show this with a particular emphasis on illustrating our ideas with examples. More technical details and proofs can be found in [5].

Since a complex conjecture with nested function symbols can have many non-variable subterm positions where Expand is applicable we make use of *inductively complete positions*, which are a special case of the general definition in [6]. This enables us to select a position in a nested term to which the computation of critical pairs can be restricted. This notion is not needed for simple conjectures since for them there is only one position on which critical pairs can be computed.

Definition 26 (Inductively Complete Positions). *A position p in a term t is* inductively complete *if $t|_p = \mathsf{f}(q^*)$, where $\mathsf{f} \in \mathcal{F}_d$ and $q^* \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$.*

From now on, the inference rule Expand is replaced by the inference rule Expand' given in Figure 2.

$$\textsf{Expand'}\ \frac{\mathcal{E} \cup \{r_1 \stackrel{\cdot}{\approx} r_2\}, \mathcal{H}}{\mathcal{E} \cup \mathcal{E}', \mathcal{H} \cup \{r_1 \to r_2\}} \quad \begin{array}{l}\text{if } r_1 \succ r_2 \text{ and } \mathcal{E}' = \mathcal{CP}'(\mathcal{R}, r_1 \to r_2, p) \\ \text{for an inductively complete position } p \text{ in } r_1\end{array}$$

Fig. 2. The inference rule Expand'

A function g is *jointly compatible* with a set of functions F on argument j if in any term $\mathsf{g}(\ldots, \mathsf{f}(\ldots), \ldots)$, where $\mathsf{f}(\ldots)$ is on the j^{th} argument of g and $\mathsf{f} \in \mathsf{F}$, every context created by rewriting f will move outside the term by rewriting g.

Definition 27 (Jointly Compatible Functions). *Let $\mathsf{F} = \{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ be jointly $\mathcal{T}$-based, let g be $\mathcal{T}$-based, let $1 \le j \le m = \mathsf{arity}(\mathsf{g})$. Then g is* jointly compatible with F on argument j *if all rules $\mathsf{f}_i(s^*) \to C[\mathsf{f}_{i_1}(t_1^*), \ldots, \mathsf{f}_{i_n}(t_n^*)]$ satisfy*

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

for a context D over $\mathcal{F}_\mathcal{T}$, $i_1, \ldots, i_k \in \{1, \ldots, n\}$, and $z_i \notin \mathcal{V}(D)$ for all $1 \le i \le n$.

The notion of compatibility in [8] is a special case of this definition. If $\mathsf{F} = \{\mathsf{f}\}$ we also say that g is compatible with f.

Example 28. Consider this TRS over $\mathcal{T}_C$.

$$\begin{array}{lll} 1 : \mathsf{zip}(x, \mathsf{nil}) \to \mathsf{pnil} & 2 : \mathsf{zip}(\mathsf{nil}, \mathsf{cons}(x', y')) & \to \mathsf{pnil} \\ 3 : \mathsf{zip}(\mathsf{cons}(x, y), \mathsf{cons}(x', y')) & \to \mathsf{pcons}(\mathsf{pair}(x, x'), \mathsf{zip}(y, y')) & \\ 4 : \mathsf{fst}(\mathsf{pnil}) \to \mathsf{nil} & 5 : \mathsf{fst}(\mathsf{pcons}(\mathsf{pair}(x, x'), y)) & \to \mathsf{cons}(x, \mathsf{fst}(y)) \end{array}$$

Then, fst is compatible with zip on argument 1. For rules 1 and 2, C is pnil (a context without holes), and $\mathsf{fst}(\mathsf{pnil})$ rewrites to nil using rule 4, i.e., $D = \mathsf{nil}$. For rule 3, C is $\mathsf{pcons}(\mathsf{pair}(x, x'), \Box)$ and $\mathsf{fst}(\mathsf{pcons}(\mathsf{pair}(x, x'), z_1))$ rewrites to $\mathsf{cons}(x, \mathsf{fst}(z_1))$ by rule 5, i.e., $D = \mathsf{cons}(x, \Box)$. ◊

As in [15,8], the concept of compatibility can be extended to arbitrarily deep nestings. To this end, we define the notion of a *compatibility sequence*.

Definition 29 (Joint Compatibility Sequences). *Let $L = \{l_1, \ldots, l_n\}$ be a set of terms in $\mathcal{T}erms(\mathcal{F}, \mathcal{V})$, let $\mathsf{g}_1, \ldots, \mathsf{g}_d$ be $\mathcal{T}$-based for some $d \ge 0$, and let the set of function symbols $\mathsf{F} = \{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ be jointly $\mathcal{T}$-based. The sequence $\langle \mathsf{g}_1, \ldots, \mathsf{g}_d, \mathsf{F} \rangle$ is a* joint compatibility sequence on arguments $\langle j_1, \ldots, j_d \rangle$ *and the set $L = \{l_1, \ldots, l_n\}$ of terms* has *this joint compatibility sequence if*

1. *g_i is compatible with g_{i+1} on argument j_i for all $1 \le i \le d-1$, and g_d is jointly compatible with F, and*

2. $l_k = \mathsf{g}_1(p_1^*, \mathsf{g}_2(p_2^*, \ldots \mathsf{g}_d(p_d^*, \mathsf{f}_k(x_k^*), q_d^*) \ldots, q_2^*), q_1^*)$, *where the* $x_{k,i}$ *do not occur elsewhere in* l_k, *all* $p_i^*, q_i^* \in \mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, *and* $\mathsf{g}_i(p_i^*, \mathsf{g}_{i+1}(\ldots), q_i^*)|_{j_i} = \mathsf{g}_{i+1}(\ldots)$ *for all* $1 \leq i \leq d-1$, *as well as* $\mathsf{g}_d(p_d^*, \mathsf{f}_k(x_k^*), q_d^*)|_{j_d} = \mathsf{f}_k(x_k^*)$.

Informally, this definition ensures that the l_i are constructed using (jointly) compatible functions in the appropriate positions.

Again, the compatibility sequences of [8] are a special case of this definition. If $L = \{l\}$ (and thus $\mathsf{F} = \{\mathsf{f}\}$) we also say that l has a compatibility sequence where we write f instead of F in the last component.

If l_i is as in the definition, then the position $p = j_1.j_2.\cdots.j_d$ is an inductively complete position in l_i.

Definition 30 (Conjunctive Complex Conjectures). *A* conjunctive complex conjecture *is a conjecture of the form* $l_1 \approx r_1 \wedge \ldots \wedge l_n \approx r_n$ *such that* $l_i \succ r_i$ *for all* $1 \leq i \leq n$, *where the* r_i *are in* $\mathcal{T}erms(\mathcal{F}_\mathcal{T}, \mathcal{V})$, *the* $l_i = D[\mathsf{f}_i(x_i^*)]$ *have a joint compatibility sequence, the set* $\mathsf{F} = \{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ *is jointly* $\mathcal{T}$*-based, and if* $x_{i,l_1} = x_{i,l_2}$ *then exists* $\langle \mathsf{f}_i, l_1, l_2, \mathcal{C}\rangle \in \mathcal{I}mp\mathcal{E}q(\mathsf{F})$ *such that* $x_{j,k_1} = x_{j,k_2}$ *for all* $\langle \mathsf{f}_j, k_1, k_2\rangle \in \mathcal{C}$.

If the group of joint $\mathcal{T}$-based functions consists of a single function, then we call a conjecture as defined above *complex*, but in contrast to the complex conjectures of [15] we allow for nonlinear complex conjectures.

The following theorem extends the results from the previous two sections to conjunctive complex conjectures and gives a decision procedure for this class.

Theorem 31. *The inductive validity of a conjunctive complex conjecture is decidable using the strategy* $\mathsf{Expand'}^* \cdot \mathsf{Simplify}^* \cdot (\mathsf{Theory}_1 \cup \mathsf{Theory}_2)^*$, *where* Expand' *is applied once to each equation of the conjecture at the innermost position, i.e., at the* f_i *for all* $1 \leq i \leq n$.

Example 32. Continuing Example 28, the term $\mathsf{fst}(\mathsf{zip}(x'', x''))$ has the compatibility sequence $\langle \mathsf{fst}, \mathsf{zip}\rangle$ on arguments $\langle 1\rangle$. Furthermore we have $\langle 1, 2, \{\langle 1, 2\rangle\}\rangle \in \mathcal{I}mp\mathcal{E}q(\mathsf{zip})$. Thus, $\mathsf{fst}(\mathsf{zip}(x'', x'')) \approx x''$ is a complex conjecture. Due to the nonlinearity, it is not permitted in [15]. ◊

Example 33. We consider mutually recursive functions in this example. Take the function fst defined in Example 28, and add the following rules defining stitch, where pair is a new constructor.

$$\begin{array}{ll} \mathsf{stitch}(x, \mathsf{nil}) \to \mathsf{pnil} & \mathsf{stitch}(\mathsf{nil}, \mathsf{cons}(x', y')) \to \mathsf{pnil} \\ \mathsf{stitch}(\mathsf{cons}(x, y), \mathsf{cons}(x', y')) \to \mathsf{pcons}(\mathsf{pair}(x, x'), \mathsf{stitch}'(y, y')) & \\ \mathsf{stitch}'(x, \mathsf{nil}) \to \mathsf{pnil} & \mathsf{stitch}'(\mathsf{nil}, \mathsf{cons}(x', y')) \to \mathsf{pnil} \\ \mathsf{stitch}'(\mathsf{cons}(x, y), \mathsf{cons}(x', y')) \to \mathsf{pcons}(\mathsf{pair}(x', x), \mathsf{stitch}(y, y')) & \end{array}$$

Then $\mathsf{F} = \{\mathsf{stitch}, \mathsf{stitch}'\}$ is jointly $\mathcal{T}$-based and fst is jointly compatible with F, since for the third stitch-rule, the term $\mathsf{fst}(\mathsf{pcons}(\mathsf{pair}(x, x'), z_1))$ rewrites to $\mathsf{cons}(x, \mathsf{fst}(z_1))$, and similarly for the third stitch'-rule. Furthermore, the set $L = \{\mathsf{fst}(\mathsf{stitch}(x'', x'')), \mathsf{fst}(\mathsf{stitch}'(y'', y''))\}$ has the joint compatibility sequence

$\langle \mathsf{fst}, \{\mathsf{stitch}, \mathsf{stitch}'\}\rangle$ on arguments $\langle 1\rangle$. Thus, the conjecture $\mathsf{fst}(\mathsf{stitch}(x'', x'')) \approx x'' \wedge \mathsf{fst}(\mathsf{stitch}'(y'', y'')) \approx y''$ is a conjunctive complex conjecture since also both $\langle \mathsf{stitch}, 1, 2, \{\langle \mathsf{stitch}', 1, 2\rangle\}\rangle$ and $\langle \mathsf{stitch}', 1, 2, \{\langle \mathsf{stitch}, 1, 2\rangle\}\rangle$ are in $\mathcal{I}mp\mathcal{E}q(\mathsf{F})$. $\Diamond$

For a conjunctive complex conjectures, $\mathcal{TP}at(\mathsf{F})$ can only be used if condition (2.) in Definition 13 is not needed. The class of *conjunctive generalized complex conjectures* is obtained from Definition 30 by replacing $\mathcal{I}mp\mathcal{E}q$ with $\mathcal{TP}at$ using this restriction.

Example 34. Consider the TRS defining zip and fst from Example 28 and the conjecture $\mathsf{fst}(\mathsf{zip}(y'', \mathsf{cons}(x'', y''))) \approx y''$. Firstly, $(y'', \mathsf{cons}(x'', y'')) \in \mathcal{TP}at(\mathsf{zip})$. For rule 3, we get $\mathcal{CU}_{\mathcal{T}_C}(\{y'' \approx^?_{\mathcal{T}_C} \mathsf{cons}(x, y), \mathsf{cons}(x'', y') \approx^?_{\mathcal{T}_C} \mathsf{cons}(x', y')\}) = \{\sigma\}$, where $\sigma = \{x' \mapsto x'', y' \mapsto \mathsf{cons}(x, y), y'' \mapsto \mathsf{cons}(x, y)\}$. Now, $\mathsf{zip}(y, y')\sigma = \mathsf{zip}(y, \mathsf{cons}(x, y))$, and $(y, \mathsf{cons}(x, y))$ is $\mathcal{T}_C$-matched by $(y'', \mathsf{cons}(x'', y''))$. Thus, the conjecture is generalized complex. $\Diamond$

7 Conclusion and Further Work

This paper shows how implicit induction methods can be used to integrate inductive reasoning into decision procedures without losing automation. We have given decision procedures based on implicit induction methods for the inductive validity of large classes of simple and complex conjectures about recursively defined theory-based functions, satisfying certain conditions that are checkable syntactically or using the decision procedure for the underlying theory.

We have broadened the class of decidable inductive conjectures permitted in [15] significantly by allowing nonlinear conjectures as well as general terms from the decidable theory on inductive positions. We have extended the notion of theory-based functions to allow for recursive calls to other function symbols as long as all the functions are being defined together. This extension allows us to decide inductive properties of mutually recursive functions automatically.

In [8], the class of conjectures whose inductive validity can be decided is expanded to linear conjectures having defined function symbols on both sides. We believe that this class can be decided using implicit induction methods as well (if the conjecture can be oriented). However, this possibility needs to be investigated further. Then it should also be possible to extend the class in [8] to non-linear conjectures and jointly theory-based functions.

It is shown in this paper that a conjunction of conjectures about all jointly theory-based definitions has to be considered simultaneously. We are confident that it is possible to decide a conjecture about a single function in a set of jointly defined functions by automatically generating conjectures about other functions in the set which are needed in a proof attempt. This might require techniques similar to the ones in [16]. We are planning to examine this idea further.

A preliminary implementation and evaluation of the results given in this paper for $\mathcal{T}_C$ is available at `http://www.cs.unm.edu/~spf/sail/`. On average, checking whether a conjecture satisfies the conditions that make it decidable takes half the time of the actual proof attempt.

Acknowledgments. We thank Mahadevan Subramaniam and the anonymous referees for useful comments.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
2. L. Bachmair. Proof by consistency in equational theories. In *Proc. LICS '88*, pages 228–233, 1988.
3. T. Ball and S. K. Rajamani. The SLAM toolkit. In *Proc. CAV '01*, LNCS 2102, pages 260–264, 2001.
4. A. Bouhoula. Automated theorem proving by test set induction. *Journal of Symbolic Computation*, 23(1):47–77, 1997.
5. S. Falke and D. Kapur. Implicit induction methods and decision procedures. Technical Report TR-CS-2006-04, Department of Computer Science, University of New Mexico, available at `http://www.cs.unm.edu/research/`, 2006.
6. L. Fribourg. A strong restriction of the inductive completion procedure. *Journal of Symbolic Computation*, 8(3):253–276, 1989.
7. J. Giesl and D. Kapur. Decidable classes of inductive theorems. In *Proc. IJCAR '01*, LNCS 2083, pages 469–484, 2001.
8. J. Giesl and D. Kapur. Deciding inductive validity of equations. In *Proc. CADE '03*, LNAI 2741, pages 17–31, 2003.
9. T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Software verification with BLAST. In *Proc. SPIN '03*, LNCS 2648, pages 235–239, 2003.
10. G. P. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and Systems Sciences*, 25(2):239–266, 1982.
11. J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1):1–33, 1989.
12. D. Kapur, J. Giesl, and M. Subramaniam. Induction and decision procedures. *Revista de la Real Academia de Ciencias, Serie A*, 98(1):153–180, 2004.
13. D. Kapur and D. R. Musser. Proof by consistency. *Artificial Intelligence*, 31(2):125–157, 1987.
14. D. Kapur, P. Narendran, and H. Zhang. Automating inductionless induction using test sets. *Journal of Symbolic Computation*, 11(1–2):81–111, 1991.
15. D. Kapur and M. Subramaniam. Extending decision procedures with induction schemes. In *Proc. CADE '00*, LNCS 1831, pages 324–345, 2000.
16. D. Kapur and M. Subramaniam. Automatic generation of simple lemmas from recursive definitions using decision procedures—preliminary report. In *Proc. ASIAN '03*, LNCS 2896, pages 125–145, 2003.
17. W. Küchlin. Inductive completion by ground proof transformation. In *Resolution of Equations in Algebraic Structures Vol. 2*, pages 211–244. Academic Press, 1989.
18. D. R. Musser. On proving inductive properties of abstract data types. In *Proc. POPL '80*, pages 154–162, 1980.
19. U. S. Reddy. Term rewriting induction. In *Proc. CADE '90*, LNCS 449, pages 162–177, 1990.
20. H. Zhang, D. Kapur, and M. S. Krishnamoorthy. A mechanizable induction principle for equational specifications. In *Proc. CADE '88*, LNCS 310, pages 162–181, 1988.