Foundations of Software Technology and Theoretical Computer Science (Bangalore) 2008.
Editors: R. Hariharan, M. Mukund, V. Vinay; pp 13-24

Sound Lemma Generation for Proving Inductive Validity of Equations

Takahito Aoto
RIEC, Tohoku University
aoto@nue.riec.tohoku.ac.jp

ABSTRACT. In many automated methods for proving inductive theorems, finding a suitable generalization of a conjecture is a key for the success of proof attempts. On the other hand, an obtained generalized conjecture may not be a theorem, and in this case hopeless proof attempts for the incorrect conjecture are made, which is against the success and efficiency of theorem proving. Urso and Kounalis (2004) proposed a generalization method for proving inductive validity of equations, called sound generalization, that avoids such an over-generalization. Their method guarantees that if the original conjecture is an inductive theorem then so is the obtained generalization. In this paper, we revise and extend their method. We restore a condition on one of the characteristic argument positions imposed in their previous paper and show that otherwise there exists a counterexample to their main theorem. We also relax a condition imposed in their framework and add some flexibilities to some of other characteristic argument positions so as to enlarge the scope of the technique.

1 Introduction

Reasoning on data structures or recursively defined domains is very common in formal treatments of programs such as program verification and program transformation. Such a reasoning often needs highly use of induction, that is, the properties of interest are not only (general) theorems which hold in all models of the theory but *inductive theorems* which hold only in a particular model, the initial model of the theory.

Although automated reasoning of inductive theorems has been investigated in many years, comparing to the high degree of automation on automated proving of (general) theorems, automated proving of inductive theorems is still considered as a very challenging problem [8]. Many approaches to automated proving of inductive theorems are known: explicit induction with sophisticated heuristics and/or decision procedures [4, 5, 6, 11, 13, 17], *implicit* induction methods such as inductionless induction/coverset induction/rewriting induction [3, 7, 9, 14, 16, 19].

In all these approaches, it is commonly understood that an introduction of suitable lemmas is an important key for the success of proof attempts. Thus techniques for finding suitable lemmas in the course of proof attempts have been investigated [12, 15, 18, 21, 22]. Among them, one of the most basic methods is *generalization*—replacing some of equivalent subterms of the conjecture by a fresh variable. Proving generalized conjecture is often easier than the original conjecture because generalization often suppress the complexity at the induction step and sometimes makes another induction scheme possible. On the other hand, the generalized conjecture may not be a theorem any more—this phenomenon is often refereed to as *over-generalization*. Because hopeless proof attempts for the incorrect conjecture is against the success and efficiency of theorem proving, any over-generalization is always better to be avoided.

Urso and Kounalis [21] proposed a generalization method called *sound generalization*, which avoid such an over-generalization in automated inductive theorem proving of equations. Their method is sound in the sense it guarantees that if the original conjecture is an inductive theorem then so is the obtained generalization. Thus the original conjecture can be safely replaced by the obtained generalization if the criteria is satisfied. However, the paper [21] contains an incorrect proof and, in fact, there exists a counterexample to their main theorem.

Example 1 (counterexample) *Let* $\mathcal{S} = \{\text{Nat}\}$, $\mathcal{F} = \{\ \mathsf{plus}^{\text{Nat}\times\text{Nat}\to\text{Nat}}, \mathsf{f}^{\text{Nat}\to\text{Nat}}, \mathsf{s}^{\text{Nat}\to\text{Nat}}, \mathsf{0}^{\text{Nat}}\ \}$ *and*

$$\mathcal{R} = \left\{ \begin{array}{lcl} \mathsf{plus}(\mathsf{0}, y) & \rightarrow & y \\ \mathsf{plus}(\mathsf{s}(x), y) & \rightarrow & \mathsf{s}(\mathsf{plus}(x, y)) \\ \mathsf{f}(\mathsf{0}) & \rightarrow & \mathsf{s}(\mathsf{s}(\mathsf{0})) \\ \mathsf{f}(\mathsf{s}(x)) & \rightarrow & \mathsf{s}(\mathsf{s}(x)) \end{array} \right\}$$

Then $\mathcal{R}$ *is a monomorphic TRS and the argument* 1 *is a downward position of* f *[21]. In a mathematical notation,* f *is a function* f *like this:*

$$f(x) = \begin{cases} 2 & \textit{if } x = 0 \\ x+1 & \textit{if } x > 0 \end{cases}$$

Let $s \equiv \mathsf{s}(\mathsf{f}(\mathsf{plus}(x, \mathsf{s}(\mathsf{0}))))$, $t \equiv \mathsf{f}(\mathsf{s}(\mathsf{plus}(x, \mathsf{s}(\mathsf{0}))))$ *and consider a conjecture* $s \doteq t$*, i.e.*

$$\mathsf{s}(\mathsf{f}(\mathsf{plus}(x, \mathsf{s}(\mathsf{0})))) \doteq \mathsf{f}(\mathsf{s}(\mathsf{plus}(x, \mathsf{s}(\mathsf{0}))))$$

Then clearly this is an inductive theorem (on natural numbers), since we have $f(x+1)+1 = (x+2)+1 = f(x+2)$*. Now let us try a generalization of this conjecture based on the original sound generalization [21]. We have* $1.1.1 = \mathrm{BP}(s)$ *and* $1.1.1 = \mathrm{BP}(t)$ *and thus* 1.1 *is a bottom path of* s *and* t*. Since* $\mathrm{bot}(s, 1.1) \equiv \mathsf{plus}(x, \mathsf{s}(\mathsf{0})) \equiv \mathrm{bot}(t, 1.1)$ *and* $s/1.1. \equiv \mathsf{plus}(x, \mathsf{s}(\mathsf{0})) \equiv t/1.1$*, the generalization at* 1.1 *in* s *and at* 1.1 *in* t *is possible. Hence we obtain a generalized conjecture*

$$\mathsf{s}(\mathsf{f}(y)) \doteq \mathsf{f}(\mathsf{s}(y))$$

However, this is not an inductive theorem since $\mathsf{s}(\mathsf{f}(\mathsf{0})) \rightarrow_{\mathcal{R}} \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0})))$ *and* $\mathsf{f}(\mathsf{s}(\mathsf{0})) \rightarrow_{\mathcal{R}} \mathsf{s}(\mathsf{s}(\mathsf{0}))$*. Therefore, this generalization is not sound contrary to the* ***Theorem 37*** *of [21].*

The purpose of this paper is to correct and extend the sound generalization proposed in [21]. In the sound generalization, generalizable subterms are computed based on five types of argument positions of functions—namely, reflective argument, downward position, upward position, down-contextual position, and up-contextual position when the term rewriting system is monomorphic. We clarify that the notion of downward position should be weakened as in their previous paper [20] that proposes induction on term partition, otherwise there is a counterexample (as presented above) and that the notion of down-contextual and up-contextual position can be enlarged so that more flexible rewrite rules are allowed for functions to have such positions. We relax the definition of monomorphic signature and localize the monomorphic and left-linearity conditions so as to enlarge the scope of the sound generalization.

The rest of the paper is organized as follows. After fixing basic notation (Section 2), we introduce a relaxed definition of monomorphic signature and revised definitions of argument positions and prove the characterization lemmas for these argument positions (Section 3). The term partition and sound generalization techniques are presented in Section 4. Section 5 concludes.

2 Preliminaries

We assume familiarity with basic notations on (many-sorted) term rewriting ([2, 10]).

Let $\mathcal{S}$ be a set of sorts and $\mathcal{F}$ be a set of $\mathcal{S}$-sorted function symbols. We assume there is a function from $\mathcal{F}$ to $\mathcal{S}^* \times \mathcal{S}$, denoted by sort. For $f \in \mathcal{F}$, let $\mathrm{sort}(f) = \langle \tau_1 \cdots \tau_n, \tau_0 \rangle$. Then $\langle \tau_1 \cdots \tau_n, \tau_0 \rangle$ is called the sort of f and denoted by $\tau_1 \times \cdots \times \tau_n \to \tau_0$. If $n = 0$, we write $\mathrm{sort}(f) = \tau$ and f is called a *constant* of sort τ.

Let V^τ be the set of variables of sort $\tau \in \mathcal{S}$. We assume there is a countably infinite set V^τ of variables for each $\tau \in \mathcal{S}$. We denote by V the set $\bigcup_{\tau \in \mathcal{S}} V^\tau$. The set $\mathrm{T}(\mathcal{F}, V)^\tau$ of terms of sort $\tau \in \mathcal{S}$ over $\mathcal{F}, V$ is defined inductively as: (1) $V^\tau \subseteq \mathrm{T}(\mathcal{F}, V)^\tau$; (2) if $f \in \mathcal{F}$, $\mathrm{sort}(f) = \tau_1 \times \cdots \times \tau_n \to \tau_0$ $(n \geq 0)$, $t_i \in \mathrm{T}(\mathcal{F}, V)^{\tau_i}$ for $1 \leq i \leq n$, then $f(t_1, \ldots, t_n) \in \mathrm{T}(\mathcal{F}, V)^{\tau_0}$. We denote by $\mathrm{T}(\mathcal{F}, V)$ the set $\bigcup_{\tau \in \mathcal{S}} \mathrm{T}(\mathcal{F}, V)^\tau$. We write t^τ if $t \in \mathrm{T}(\mathcal{F}, V)^\tau$. The set of variables contained in a term t is denoted by $\mathrm{V}(t)$. We use $\equiv$ to denote the syntactical equality.

A *position* is a (possibly empty) sequence of positive integers. The empty sequence is denoted by ϵ and the concatenation of positions p and q is by $p.q$. The set $\mathrm{Pos}(t)$ of positions (or *occurrence*) in a term t and the *subterm* t/p of t at the position p are recursively defined as follows: for $t \in V$, $\mathrm{Pos}(t) = \{\epsilon\}$ and $t/\epsilon = t$; for $t \equiv f(t_1, \ldots, t_n)$, $\mathrm{Pos}(t) = \{\epsilon\} \cup \bigcup_{1 \leq i \leq n} \{i.p \mid p \in \mathrm{Pos}(t_i)\}$, $t/\epsilon = t$, and $t/i.p = t_i/p$. If $p \in \mathrm{Pos}(t)$ and $\mathrm{sort}(t/p) = \mathrm{sort}(s)$, we write $t[s]_p$ the term obtained from t by replacing the subterm with s at the position p. A variable $x \in \mathrm{V}(t)$ is said to have a *linear variable occurrence* in t if there exists a unique $p \in \mathrm{Pos}(t)$ such that $x \equiv t/p$. The prefix ordering $\leq$ on positions are defined as $p \leq q$ iff $q = p.r$ for some position r. We write $p \mid q$ if neither $p \leq q$ nor $q \leq p$ hold. A set of position P is said to be *prefix-closed* if $p \in P$ and $q \leq p$ imply $q \in P$. The function symbol that occurs in t at a position $p \in \mathrm{Pos}(t)$ is denoted by $t(p)$. In particular, the *root symbol* of a term t is $t(\epsilon)$.

Suppose $\Box^\tau$ is a constant of sort τ and $\{\Box^\tau \mid \tau \in \mathcal{S}\} \cap \mathcal{F} = \emptyset$. A *context* is an element in $\mathrm{T}(\mathcal{F} \cup \{\Box^\tau \mid \tau \in \mathcal{S}\}, V)$. The special constants $\Box^\tau$ are called *holes*. If the holes occurring in a context C are $\Box^{\tau_1}, \ldots, \Box^{\tau_n}$ from left to right and $t_1, \ldots, t_n$ are terms of sorts $\tau_1, \ldots, \tau_n$, respectively, then we denote by $C[t_1, \ldots, t_n]$ the term obtained by replacing the holes $\Box^{\tau_1}, \ldots, \Box^{\tau_n}$ with the terms $t_1, \ldots, t_n$. The superscript of holes is often omitted if no confusion arises. For a position p, we write $C[u]_p$ if $C/p \equiv \Box$.

A map σ from V to $\mathrm{T}(\mathcal{F}, V)$ is called a *substitution* if (1) σ preserves sort, i.e. $\mathrm{sort}(x) = \mathrm{sort}(\sigma(x))$ and (2) the domain of σ is finite, where the domain of σ is given by $\mathrm{dom}(\sigma) = \{x \in V \mid \sigma(x) \not\equiv x\}$. A substitution σ such that $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$ and $\sigma(x_i) \equiv t_i$ $(1 \leq i \leq n)$ is also written as $\{x_1 := t_1, \ldots, x_n := t_n\}$. We identify the substitution σ and its homomorphic extension. A term $\sigma(t)$ is called an *instance* of the term t; $\sigma(t)$ is also written as $t\sigma$.

A pair $\langle l, r \rangle$ of terms l, r satisfying conditions (1) $l(\epsilon) \in \mathcal{F}$ and (2) $\mathrm{V}(r) \subseteq \mathrm{V}(l)$ (3) $\mathrm{sort}(l) =$

sort(r) is said to be a *rewrite rule*. A rewrite rule $\langle l, r\rangle$ is denoted by $l \to r$. A tuple $\langle \mathcal{S}, \mathcal{F}, \mathcal{R}\rangle$ is a *term rewriting system (TRS)*. If no confusion arises, $\langle \mathcal{S}, \mathcal{F}, \mathcal{R}\rangle$ is abbreviated as $\mathcal{R}$. If there exist a position p, a substitution σ, and a rewrite rule $l \to r \in \mathcal{R}$ such that $s/p \equiv l\sigma$ and $t \equiv s[r\sigma]_p$, we write $s \to_{\mathcal{R}} t$. We call $s \to_{\mathcal{R}} t$ a *rewrite step*, p a *redex occurrence*, and $\to_{\mathcal{R}}$ the rewrite relation of the TRS $\mathcal{R}$. The reflexive transitive closure and equivalence closure of $\to_{\mathcal{R}}$ are denoted by $\xrightarrow{*}_{\mathcal{R}}$ and $\xleftrightarrow{*}_{\mathcal{R}}$, respectively. A TRS $\mathcal{R}$ is *terminating* if $\to_{\mathcal{R}}$ is noetherian i.e. there is no infinite sequence $t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots$; is *confluent* if $\xleftarrow{*}_{\mathcal{R}} \circ \xrightarrow{*}_{\mathcal{R}} \subseteq \xrightarrow{*}_{\mathcal{R}} \circ \xleftarrow{*}_{\mathcal{R}}$. A term is said to be *normal* if there exists no s such that $t \to_{\mathcal{R}} s$. Any normal term s such that $t \xrightarrow{*}_{\mathcal{R}} s$ is called a *normal form* of t. One can easily show that if a TRS $\mathcal{R}$ is terminating and confluent, any term s has a unique normal form; the normal form of s is denoted by $s{\downarrow}_{\mathcal{R}}$, or simply by $s{\downarrow}$ if no confusion arises.

The set of *defined function symbols* is given by $\mathcal{D}_{\mathcal{R}} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$ and the set of *constructor symbols* by $\mathcal{C}_{\mathcal{R}} = \mathcal{F} \setminus \mathcal{D}_{\mathcal{R}}$. The set of defined symbols appearing in a term t is denoted by $\mathcal{D}_{\mathcal{R}}(t)$. If $\mathcal{R}$ is obvious from its context, we omit the subscript $_{\mathcal{R}}$ from $\mathcal{D}_{\mathcal{R}}, \mathcal{C}_{\mathcal{R}}$. Terms in $\mathrm{T}(\mathcal{C}, V)$ are said to be *constructor terms*.

An *equation* $l \doteq r$ is a pair $\langle l, r\rangle$ of terms of the same sort. When we write $l \doteq r$, however, we do not distinguish $\langle l, r\rangle$ and $\langle r, l\rangle$. A term t is said to be *ground* if $\mathrm{V}(t) = \emptyset$. The set of ground terms is denoted by $\mathrm{T}(\mathcal{F})$. If $t\sigma \in \mathrm{T}(\mathcal{F})$, $t\sigma$ is called a *ground instance* of t. The ground instance of an equation is defined analogously. A *ground substitution* is a substitution σ_g such that $\sigma_g(x) \in \mathrm{T}(\mathcal{F})$ for any $x \in \mathrm{dom}(\sigma_g)$. Without loss of generality, we assume that $t\sigma_g$ is ground (i.e. $\mathrm{V}(t) \subseteq \mathrm{dom}(\sigma_g)$) when we speak of an instance $t\sigma_g$ of t by a ground substitution σ_g; and so for ground instances of equations. An *inductive theorem* of a TRS $\mathcal{R}$ is an equation that is valid on $\mathrm{T}(\mathcal{F})$, i.e. $s \doteq t$ is an inductive theorem if $s\sigma_g \xleftrightarrow{*}_{\mathcal{R}} t\sigma_g$ holds for any ground instance $s\sigma_g \doteq t\sigma_g$. We write $\mathcal{R} \vdash_{ind} s \doteq t$ if $s \doteq t$ is an inductive theorem. A TRS $\mathcal{R}$ is said to be *sufficiently complete* if for any ground term $t_g \in \mathrm{T}(\mathcal{F})$, there exists a constructor ground term $s_g \in \mathrm{T}(\mathcal{C})$ such that $t_g \xleftrightarrow{*}_{\mathcal{R}} s_g$. One can easily show that if the TRS is sufficiently complete, terminating, and confluent then the normal form of any ground term is a constructor term.

Throughout this paper, we only deal with the TRSs that are sufficiently complete, terminating, and confluent.

3 Characterization of Monomorphic Equations

In this section, we introduce a relaxed definition of monomorphic signature and revised definitions of argument positions—reflective argument position, downward and upward argument positions, and contextual positions—and present lemmas that characterize these positions.

The notion of monomorphic signature is introduced by Urso and Kounalis [20, 21]. We here generalize the notion to monomorphic sorts, terms, etc.

DEFINITION 1.*[monomorphic sort]*

1. *A sort τ is said to be* monomorphic *if (i) there is only one constructor constant of the sort τ (denoted by $\perp^{\tau}$), (ii) for each non-constant constructor $g \in \mathcal{C}$ of sort $\tau_1 \times \cdots \times \tau_n \to \tau$, there exists a unique $1 \le i \le n$ such that $\tau_i = \tau$; such i is called the reflective*

argument position of g and denoted by $RA(g)$.

2. *A variable, term, equation, and rule are said to be monomorphic if its sort is monomorphic.*

Intuitively, a sort is monomorphic if each normal term of that sort has a list structure. For example, NatList (with nil : NatList and cons : Nat $\times$ NatList $\rightarrow$ NatList), Nat (with 0 : Nat and s : Nat $\rightarrow$ Nat) are monomorphic while Tree, Bool are not.

We here removed one of the conditions contained in the original definition of monomorphicness. Let $\succ_{\mathcal{S}}$ be a relation on $\mathcal{S}$ given by $\tau \succ_{\mathcal{S}} \rho$ iff there exists a ground constructor term $s_g[u_g^\rho]^\tau$ with $\tau \neq \rho$. In the original definition, the monomorphic signature is the one with only monomorphic sorts such that there are no ρ, δ such that $\rho \succ_{\mathcal{S}} \delta \succ_{\mathcal{S}} \rho$. The acyclicity of $\succ_{\mathcal{S}}$, however, turns out to be unnecessary in the subsequent development for sound generalization. Moreover, the monomorphic condition can be localized so that the signature may contain non-monomorphic sorts as well. This relaxation is useful, for example, to deal with BoolList.

We introduce a notion of reflective positions in a monomorphic term as a successive sequence of reflective argument positions from its root. Then, based on this, we define a join operator. This is in contrast to the original definition in [20, 21] where the join operator is defined as the replacement with $\perp$. The elimination of the extra restriction of monomorphic signature is achieved due to our new definition.

DEFINITION 2.*[reflective position] The set* $\mathrm{RPos}(t)$ *of reflective positions in t is defined as follows: (i)* $\epsilon \in \mathrm{RPos}(t)$ *(ii) if* $t \equiv g(t_1, \ldots, t_n)$ *with* $g \in \mathcal{C}$, $i = RA(g)$, *and* $p \in \mathrm{RPos}(t_i)$ *then* $i.p \in \mathrm{RPos}(t_i)$.

For example, we have $\mathrm{RPos}(\mathsf{s}(\mathsf{s}(0))) = \{\epsilon, 1, 1.1\}$. Since $\mathrm{RPos}(t)$ is total w.r.t. the prefix ordering $\leq$, there exists a position p that is greatest (w.r.t. $\leq$) in $\mathrm{RPos}(t)$.

DEFINITION 3.*[greatest reflective position] Let t be a monomorphic term. The greatest element w.r.t. the prefix ordering in* $\mathrm{RPos}(t)$ *is called the greatest reflective position (grp) of t.*

DEFINITION 4.*[join operator] For each monomorphic sort τ, a join operator $\otimes^\tau$ on the set* $\mathrm{T}(\mathcal{C})$ *is defined as follows: for ground constructor terms s_g and t_g of sort τ, $s_g \otimes^\tau t_g = s_g[t_g]_p$ where p is the grp of s_g. We omit the superscript τ if no confusion arises.*

The following properties of join operator is easily verified.

LEMMA 5.*[properties of join operator] Let $s_g, t_g, u_g \in \mathrm{T}(\mathcal{C})$ be monomorphic terms.*

1. *If $s_g \otimes t_g \equiv s_g \otimes u_g$ then $t_g \equiv u_g$. If $s_g \otimes t_g \equiv u_g \otimes t_g$ then $s_g \equiv u_g$.*
2. $(s_g \otimes t_g) \otimes u_g \equiv s_g \otimes (t_g \otimes u_g)$.
3. $p \in \mathrm{RPos}(u_g)$ *implies* $u_g[s_g \otimes t_g]_p \equiv u_g[s_g]_p \otimes t_g$.
4. $\perp \otimes t_g \equiv t_g$ *and* $s_g \otimes \perp \equiv s_g$.

LEMMA 6.*[decomposition at a reflective position] Suppose $t_g, u_g \in \mathrm{T}(\mathcal{F})$ are monomorphic and $p \in \mathrm{RPos}(t_g)$. Then $t_g[u_g]_p\!\downarrow \equiv t_g[\perp]_p\!\downarrow \otimes u_g\!\downarrow$.*

PROOF. By induction on p. (B.S.) Suppose $p = \epsilon$. Then $t_g[u_g]_p\!\downarrow \equiv u_g\!\downarrow \equiv \perp \otimes u_g\!\downarrow \equiv \perp\!\downarrow \otimes u_g\!\downarrow \equiv t_g[\perp]_p\!\downarrow \otimes u_g\!\downarrow$. (I.S.) Let $p = i.q$ with $t_g \equiv g(t_1, \ldots, t_n)$, $g \in \mathcal{C}$, $i = RA(g)$, and

$q \in \mathrm{RPos}(t_i)$. Then

$$\begin{array}{rcll} t_g[u_g]_p{\downarrow} & \equiv & g(t_1,\dots,t_i[u_g]_q,\dots,t_n){\downarrow} & \text{by definition} \\ & \equiv & g(t_1{\downarrow},\dots,t_i[u_g]_q{\downarrow},\dots,t_n{\downarrow}) & \text{by } g \in \mathcal{C} \\ & \equiv & g(t_1{\downarrow},\dots,t_i[\bot]_q{\downarrow} \otimes u_g{\downarrow},\dots,t_n{\downarrow}) & \text{by the induction hypothesis} \\ & \equiv & g(t_1{\downarrow},\dots,t_i[\bot]_q{\downarrow},\dots,t_n{\downarrow}) \otimes u_g{\downarrow} & \text{by } i = RA(g) \text{ and } \textbf{Lemma 5} \\ & \equiv & g(t_1,\dots,t_i[\bot]_q,\dots,t_n){\downarrow} \otimes u_g{\downarrow} & \text{by } g \in \mathcal{C} \\ & \equiv & t_g[\bot]_p{\downarrow} \otimes u_g{\downarrow}. & \end{array}$$

∎

In [20, 21], the notion of downward position is defined recursively; however, the mutual recursion of the definition is not terminating and thus the downward positions may not be uniquely defined for a TRS. To make this fact explicit, we introduce a notion of downward argument map and that of compatibility of the map with a TRS.

DEFINITION 7.*[downward argument map/downward position]*

1. *A downward argument map DP is a partial map from $\mathcal{D}$ to $\mathbb{N}$ such that for any $f \in \mathrm{dom}(DP)$, if $i = DP(f)$ then (1) $1 \leq i \leq \mathrm{arity}(f)$, and (2) if $f : \tau_1 \times \cdots \times t_n \to \tau_0$ then $\tau_i = \tau_0$ and τ_0 is monomorphic.*
2. *Let p be a position in a term t. The set $\mathrm{DPos}(t)$ of downward positions in t is defined as follows: (i) $\epsilon \in \mathrm{DPos}(t)$ (ii) if $t \equiv g(t_1,\dots,t_n)$ with $g \in \mathcal{C}$, $i = RA(g)$, and $p \in \mathrm{DPos}(t_i)$ then $i.p \in \mathrm{DPos}(t_i)$. (iii) $t \equiv f(t_1,\dots,t_n)$ with $f \in \mathcal{D}$, $i = DP(f)$, and $p \in \mathrm{DPos}(t_i)$ then $i.p \in \mathrm{DPos}(t_i)$.*

DEFINITION 8.*[compatible downward argument map] A downward argument map DP is compatible with a set $\mathcal{R}$ of rewrite rules if for any $f \in \mathrm{dom}(DP)$ with $i = DP(f)$ and for any $f(l_1,\dots,l_n) \to r \in \mathcal{R}$, l_i is a linear variable occurrence of $f(l_1,\dots,l_n)$ and there exists a position $p \in \mathrm{DPos}(r)$ such that $l_i \equiv r/p$ and r/p is a linear variable occurrence in r.*

Contrary to the definition in [21] in which l_i ($\equiv r/p$) is allowed to be an arbitrary term when $p \neq \epsilon$, we impose a restriction that l_i must be a variable; this condition is imposed in their previous paper [20] that proposes induction on term partition.

Example 2 *Let $\mathcal{S} = \{\mathrm{Nat}\}$, $\mathcal{F} = \{\ \mathsf{plus}^{\mathrm{Nat}\times\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{s}^{\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{0}^{\mathrm{Nat}}\ \}$ and*

$$\mathcal{R} = \left\{ \begin{array}{lcl} \mathsf{plus}(\mathsf{0},y) & \to & y \\ \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y)) \end{array} \right\}.$$

Then $\bot^{\mathrm{Nat}} \equiv \mathsf{0}$ and $\mathrm{RA}(\mathsf{s}) = 1$. A map DP with $\mathrm{DP}(\mathsf{plus}) = 2$ is a downward argument map compatible with $\mathcal{R}$.

Example 3 *Let $\mathcal{S} = \{\mathrm{Nat}\}$, $\mathcal{F} = \{\ \mathsf{f}^{\mathrm{Nat}\times\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{g}^{\mathrm{Nat}\times\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{s}^{\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{0}^{\mathrm{Nat}}\ \}$ and*

$$\mathcal{R} = \left\{ \begin{array}{lcllcl} \mathsf{f}(\mathsf{0},y) & \to & y & \mathsf{g}(x,\mathsf{0}) & \to & x \\ \mathsf{f}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{g}(y,x)) & \mathsf{g}(x,\mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{f}(y,x)) \end{array} \right\}.$$

Then functions $\{\mathsf{f} \mapsto 2, \mathsf{g} \mapsto 1\}$ and $\emptyset$ are both downward argument maps compatible with $\mathcal{R}$. In terms of [20, 21], it may possibly be (1) 2 is a downward position of f and 1 is a downward position of g, and (2) both of f and g do not have downward positions. This is why we introduced the notion of downward argument maps as remarked above.

LEMMA 9.[*preservation of a downward position*] *Suppose that* DP *is compatible with* $\mathcal{R}$. *Let* z *be a fresh variable.*

1. *Let* $p \in \mathrm{DPos}(s_g)$ *and* $s_g \to_{\mathcal{R}} t_g$. *Then either (1)* $s_g/p \equiv t_g/q$ *and* $s_g[z]_p \to_{\mathcal{R}} t_g[z]_q$ *or (2)* $p = q$, $s_g[z]_p \equiv t_g[z]_q$, *and* $s_g/p \to_{\mathcal{R}} t_g/q$.
2. *Let* $p \in \mathrm{DPos}(s_g)$ *and* $s_g \xrightarrow{*}_{\mathcal{R}} t_g$. *Then there exists* $q \in \mathrm{DPos}(t_g)$ *such that* $s_g[z]_p \xrightarrow{*}_{\mathcal{R}} t_g[z]_q$ *and* $s_g/p \xrightarrow{*}_{\mathcal{R}} t_g/q$.

PROOF. 1. Let the redex occurrence of $s_g \to_{\mathcal{R}} t_g$ be p'. If $p' \mid p$ then apparently (1) holds and if $p' \geq p$ then apparently (2) holds. It remains to show the case $p' < p$. Then there exists $f(l_1,\ldots,l_n) \to r \in \mathcal{R}$ and a substitution σ such that $s_g/p' \equiv f(l_1,\ldots,l_n)\sigma$. By $p \in \mathrm{DPos}(s_g)$, $p'.i \leq p$ with $i = DP(f)$, $l_i \equiv x \in V$ is a linear in $f(l_1,\ldots,l_n)$, and there exists a unique $u \in \mathrm{DPos}(r)$ such that $r/u \equiv x$. Then we have $p = p'.i.q'$ for some q'. Let $q = p'.u.q'$. Then $s_g/p \equiv t_g/q$. Since $p \in \mathrm{DPos}(s_g[r\sigma]_p) = \mathrm{DPos}(t_g)$, $q' \in \mathrm{DPos}(x\sigma)$, and $u \in \mathrm{DPos}(r)$, we have $q = p.u.q' \in \mathrm{DPos}(t_g)$. Furthermore, since $l_i \equiv x \in V$ and x is a linear variable in $f(l_1,\ldots,l_n)$ and r, we have $s_g[z]_p \to_{\mathcal{R}} t_g[z]_q$. 2. It follows from 1. ∎

LEMMA 10.[*decomposition at a downward position*] *Suppose that* DP *is compatible with* $\mathcal{R}$, $t_g, u_g \in \mathrm{T}(\mathcal{F})$ *are monomorphic, and* $p \in \mathrm{DPos}(t_g)$. *Then* $t_g[u_g]_p{\downarrow} \equiv t_g[\bot]_p{\downarrow} \otimes u_g{\downarrow}$.

PROOF. By **Lemma 9**, there exist s_g, v_g, q such that $t_g[u_g]_p{\downarrow} \equiv s_g[v_g]_q$, $q \in \mathrm{DPos}(s_g)$, $t_g[z]_p \xrightarrow{*}_{\mathcal{R}} s_g[z]_q$, and $u_g \xrightarrow{*}_{\mathcal{R}} v_g$. By sufficient completeness, $s_g[v_g]_q \in \mathrm{T}(\mathcal{C})$ and thus $v_g, s_g[\bot]_q \in \mathrm{T}(\mathcal{C})$ and hence $t_g[\bot]_p{\downarrow} \equiv s_g[\bot]_q$, and $u_g{\downarrow} \equiv v_g$. Furthermore, since $q \in \mathrm{DPos}(s_g)$ and $s_g[\bot]_q \in \mathrm{T}(\mathcal{C})$, it follows $q \in \mathrm{RPos}(s_g)$ by the definition of downward position. Hence, by **Lemma 6**, we have $s_g[v_g]_q \equiv s_g[\bot]_q \otimes v_g$. Therefore, $t_g[u_g]_p{\downarrow} \equiv t_g[\bot]_p{\downarrow} \otimes u_g{\downarrow}$. ∎

Example 4 (counterexample) *The lemma above does not hold for the definition of downward position in [21]. Let* $\mathcal{S} = \{\mathrm{Nat}\}$, $\mathcal{F} = \{\mathsf{f}^{\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{s}^{\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{0}^{\mathrm{Nat}}\}$, *and*

$$\mathcal{R} = \left\{ \begin{array}{lcl} \mathsf{f}(\mathsf{0}) & \to & \mathsf{s}(\mathsf{s}(\mathsf{0})) \\ \mathsf{f}(\mathsf{s}(x)) & \to & \mathsf{s}(\mathsf{s}(x)) \end{array} \right\}.$$

Then we have $\mathsf{f}(\mathsf{s}(\mathsf{0})){\downarrow} \equiv \mathsf{s}(\mathsf{s}(\mathsf{0}))$ *and* $\mathsf{f}(\mathsf{0}){\downarrow} \otimes \mathsf{s}(\mathsf{0}){\downarrow} \equiv \mathsf{s}(\mathsf{s}(\mathsf{0})) \otimes \mathsf{s}(\mathsf{0}) \equiv \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0})))$. *Thus* $\mathsf{f}(\mathsf{s}(\mathsf{0})){\downarrow} \not\equiv \mathsf{f}(\mathsf{0}){\downarrow} \otimes \mathsf{s}(\mathsf{0})$.

We now describe very roughly how the downward positions can be used to identify the common subterms that can be generalized.

Example 5 *Let* $\mathcal{S}, \mathcal{F}, \mathcal{R}$ *be as in Example 2. Consider a conjecture* e *and its generalization* e' *like this:*

$$e = \mathsf{plus}(s[x]_p, x) \doteq \mathsf{plus}(t[x]_q, x), \qquad e' = \mathsf{plus}(s[x]_p, y) \doteq \mathsf{plus}(t[x]_q, y).$$

Obviously, if the equation e' *is an inductive theorem then the equation* e *is an inductive theorem (because* e *is a particular instance of* e'*). We explain, using the decomposition at a downward position* 2, *that the other implication also holds. Suppose the equation* e *is an inductive theorem. Then, by definition,* $\mathsf{plus}(s\sigma_g[u_g]_p, u_g) \xleftrightarrow{*}_{\mathcal{R}} \mathsf{plus}(t\sigma_g[u_g]_q, u_g)$ *for any ground term* u_g *and ground substitution* σ_g. *This means* $\mathsf{plus}(s\sigma_g[u_g]_p, u_g){\downarrow} \equiv \mathsf{plus}(t\sigma_g[u_g]_q, u_g){\downarrow}$. *Thus, by* ***Lemma 10***, $\mathsf{plus}(s\sigma_g[u_g]_p, \mathsf{0})){\downarrow} \otimes u_g{\downarrow} \equiv \mathsf{plus}(t\sigma_g[u_g]_q, \mathsf{0})){\downarrow} \otimes u_g{\downarrow}$, *which implies* $\mathsf{plus}(s\sigma_g[u_g]_p, \mathsf{0})){\downarrow} \equiv$

$\mathsf{plus}(t\sigma_g[u_g]_q, 0))\downarrow$. *Then, for any* w_g, $\mathsf{plus}(s\sigma_g[u_g]_p, 0))\downarrow \otimes w_g\downarrow \equiv \mathsf{plus}(t\sigma_g[u_g]_q, 0))\downarrow \otimes w_g\downarrow$. *By* ***Lemma 10****, this implies* e' *is also an inductive theorem.*

Throughout the paper, if no confusion arises, we assume that the downward argument map DP is compatible with the TRS $\mathcal{R}$.

Next, we focus on the dual notion of downward position called upward position. The notion of upward argument position *UP* is the same as the one given in [20, 21]. We, however, additionally introduce a notion of upward position in a term which will be used to extend the definition of contextual positions.

DEFINITION 11.*[upward argument position/upward position] Let* $f \in \mathcal{D}$ *with* $f : \tau_1 \times \cdots \times \tau_n \to \tau$ *and* $1 \leq i \leq n$ *such that* $\tau_i = \tau$ *and* τ *is monomorphic.*

1. *The index i is called a upward argument position of f (*$UP(f)$*) if for any* $f(l_1, \ldots, l_n) \to r \in \mathcal{R}$*, either* $l_i \equiv \perp^\tau$ *or* $l_i \equiv u[x]_p \in \mathrm{T}(\mathcal{C}, V)$ *and* $r \equiv u[l[x]_i]_p$*, where* $l \equiv f(l_1, \ldots, l_n)$*,* $p \in \mathrm{RPos}(u)$*, and x is a linear variable in l. Note that* $p \neq \epsilon$*; for, otherwise* $l \equiv r$ *and contradicts termination of* $\mathcal{R}$*.*
2. *The set* $\mathrm{UPos}(t)$ *of upward positions in t is defined as follows:* $\mathrm{UPos}(t) = \{i\} \cup \{i.p \mid p \in \mathrm{UPos}(t_i)\}$ *if* $t \equiv f(t_1, \ldots, t_n)$ *with* $f \in D$ *and* $i = UP(f)$*;* $\mathrm{UPos}(t) = \emptyset$ *otherwise.*

The dual property of **Lemma 10** holds for upward positions.

LEMMA 12.*[decomposition at a upward position] Let* $t_g, u_g \in \mathrm{T}(\mathcal{F})$ *be monomorphic terms.*

1. *Let* $t_g \equiv f(t_1, \ldots, t_n)$ *with* $t_j \in \mathrm{T}(\mathcal{C})$ *for all* $1 \leq j \leq n$*. If* $i = UP(f)$ *and p be the grp of* t_i *then* $t_g \xrightarrow{*}_{\mathcal{R}} t_i[t_g[\perp]_i]_p$*.*
2. *If* $i = UP(t_g(\epsilon))$ *then* $t_g[u_g]_i\downarrow \equiv u_g\downarrow \otimes t_g[\perp]_i\downarrow$*.*
3. *If* $p \in \mathrm{UPos}(t_g)$ *then* $t_g[u_g]_p\downarrow \equiv u_g\downarrow \otimes t_g[\perp]_p\downarrow$*.*

PROOF.

1. By induction on $|t_i|$.
2. Use confluence, sufficient completeness of $\mathcal{R}$ and 1.
3. By induction on p. Use 2.

∎

Next, we focus on the notion of contextual argument positions. The definition is extended from the original one given in [20, 21].

DEFINITION 13.*[contextual argument position] Let* $f \in \mathcal{D}$ *with* $f : \tau_0 \times \cdots \times \tau_n \to \tau$ *with monomorphic* τ *and* $1 \leq i \leq \mathrm{arity}(f)$ *such that* τ_i *is monomorphic.*

1. *The index i is called a down-contextual argument position of f (*$DCP(f)$*) if for any* $f(l_1, \ldots, l_n) \to r \in \mathcal{R}$*, either* $l_i \equiv \perp$ *and* $r \equiv \perp$ *hold or* $l_i \equiv u[x]_p \in \mathrm{T}(\mathcal{C}, V)$ *and* $r/q \equiv l[x]_i$*, where* $l \equiv f(l_1, \ldots, l_n)$*,* $p \in \mathrm{RPos}(u)$*,* $q \in \mathrm{UPos}(r)$*, and x is a linear variable in l and r. Note that* $p \neq \epsilon$*; for, otherwise* $r/q \equiv l$ *and contradicts termination of* $\mathcal{R}$*.*
2. *The index i is called an up-contextual argument position of f (*$UCP(f)$*) if for any* $f(l_1, \ldots, l_n) \to r \in \mathcal{R}$*, either* $l_i \equiv \perp$ *and* $r \equiv \perp$ *hold or* $l_i \equiv u[x]_p \in \mathrm{T}(\mathcal{C}, V)$ *and* $r/q \equiv l[x]_i$*, where* $l \equiv f(l_1, \ldots, l_n)$*,* $p \in \mathrm{RPos}(u)$*,* $q \in \mathrm{DPos}(r)$*, and x is a linear variable in l and r. Note that* $p \neq \epsilon$*; for, otherwise* $r/q \equiv l$ *and contradicts termination of* $\mathcal{R}$*.*

The original definition of contextual positions use the conditions $q = UP(r(\epsilon))$ and $q = DP(r(\epsilon))$ instead of $q \in \mathrm{UPos}(r)$ and $q \in \mathrm{DPos}(r)$, respectively. Furthermore, when $l_i \equiv \bot$ and $r \equiv \bot$, $\mathrm{sort}(l_i) = \mathrm{sort}(r)$ (and hence $l_i \equiv r$) is required in the original definition. Since $UP(r(\epsilon)) \in \mathrm{UPos}(r)$ and $DP(r(\epsilon)) \in \mathrm{DPos}(r)$, our definition enlarges the scope of the contextual positions.

Example 6 *Let* $\mathcal{S} = \{\mathrm{Nat}, \mathrm{List}\}$, $\mathcal{F} = \{\mathsf{dbl}^{\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{len}^{\mathrm{List}\to\mathrm{Nat}}, \mathsf{sum}^{\mathrm{List}\to\mathrm{Nat}}, \mathsf{plus}^{\mathrm{Nat}\times\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{cons}^{\mathrm{Nat}\times\mathrm{List}\to\mathrm{List}}, \mathsf{nil}^{\mathrm{List}}, \mathsf{s}^{\mathrm{Nat}\to\mathrm{Nat}}, \mathsf{0}^{\mathrm{Nat}}\}$, *and*

$$\mathcal{R} = \left\{ \begin{array}{lcllcl} \mathsf{len}(\mathsf{nil}) & \to & \mathsf{0} & \mathsf{dbl}(\mathsf{0}) & \to & \mathsf{0} \\ \mathsf{len}(\mathsf{cons}(x, xs)) & \to & \mathsf{s}(\mathsf{len}(xs)) & \mathsf{dbl}(\mathsf{s}(x)) & \to & \mathsf{s}(\mathsf{s}(\mathsf{dbl}(x))) \\ \mathsf{plus}(x, \mathsf{0}) & \to & x & \mathsf{sum}(\mathsf{nil}) & \to & \mathsf{0} \\ \mathsf{plus}(x, \mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(x, y)) & \mathsf{sum}(\mathsf{cons}(x, xs)) & \to & \mathsf{plus}(x, \mathsf{sum}(xs)) \end{array} \right\}$$

Then we have $1 = UCP(\mathsf{len})$, $1 = UCP(\mathsf{dbl})$, *and* $1 = DCP(\mathsf{sum})$. *In the original definition in [20, 21], however, none of these are defined.*

LEMMA 14.*[decomposition at contextual positions] Let* $t_g \in \mathrm{T}(\mathcal{F})$, $u_g, v_g \in \mathrm{T}(\mathcal{C})$ *be monomorphic terms and* $f = t_g(\epsilon)$.

1. *If* $i = DCP(f)$ *then* $t_g[\bot]_i\!\downarrow\, \equiv \bot$.
2. *If* $i = DCP(f)$ *then* $t_g[u_g \otimes v_g]_i\!\downarrow\, \equiv t_g[v_g]_i\!\downarrow \otimes\, t_g[u_g]_i\!\downarrow$.
3. *If* $i = UCP(f)$ *then* $t_g[\bot]_i\!\downarrow\, \equiv \bot$.
4. *If* $i = UCP(f)$ *then* $t_g[u_g \otimes v_g]_i\!\downarrow\, \equiv t_g[u_g]_i\!\downarrow \otimes\, t_g[v_g]_i\!\downarrow$.

PROOF.

1. Straightforward.
2. By induction on $|u_g|$. Use **Lemma 6** and **Lemma 12**.
3. Same as 1 except using $i = UCP(f)$ instead of $i = DCP(f)$.
4. Same as 2 except using **Lemma 10** instead of **Lemma 12**.

∎

4 Term Partition and Sound Generalization

Based on the characterization of five types of argument positions, Urso and Kounalis ([20, 21]) developed techniques useful in inductive theorem proving—namely, term partition and sound generalization. These techniques rely on the following observation.

DEFINITION 15.*[term partition[20, 21]] Let* $\mathcal{R}$ *be a sufficiently complete, confluent, terminating TRS.* $\langle s_0, s_1\rangle$ *is said to be a term partition of* s *if (1)* s_0 *and* s_1 *have the same monomorphic sort* τ *and (2) for any ground substitution* θ_g, $s_0\theta_g\!\downarrow \otimes\, s_1\theta_g\!\downarrow\, \equiv s\theta_g\!\downarrow$.

PROPOSITION 16.*[term partition theorem (Theorem 1 of [20])] Let* $\mathcal{R}$ *be a sufficiently complete, confluent, terminating TRS. Suppose* $\langle s_0, s_1\rangle$ *is a term partition of* s *and* $\langle t_0, t_1\rangle$ *is a term partition of* t. *Then for each* $i \in \{0, 1\}$, *if* $\mathcal{R} \vdash_{ind} s_i \doteq t_i$ *then we have* $\mathcal{R} \vdash_{ind} s \doteq t$ *iff* $\mathcal{R} \vdash_{ind} s_{1-i} \doteq t_{1-i}$.

In [20, 21], Urso and Kounalis introduced a notion of prominent paths (called *top path* and *bottom path*) based on the five types of argument positions of functions and a method to compute some term partitions based on these paths.

DEFINITION 17.*[top/bottom paths[20, 21]] Let t be a monomorphic term. The set* $\mathrm{TPath}(t)$ *of top paths in a term t and the set* $\mathrm{BPath}(t)$ *of bottom paths in a term t are defined as follows:*

$$\mathrm{TPath}(t) = \begin{cases} \{\epsilon\} \cup \{i.p \mid p \in \mathrm{TPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = RA(f) \\ \{\epsilon\} \cup \{i.p \mid p \in \mathrm{TPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = UP(f) \\ \{\epsilon\} \cup \{i.p \mid p \in \mathrm{BPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = DCP(f) \\ \{\epsilon\} \cup \{i.p \mid p \in \mathrm{TPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = UCP(f) \\ \{\epsilon\} & \textit{otherwise} \end{cases}$$

$$\mathrm{BPath}(t) = \begin{cases} \{\epsilon\} \cup \{i.p \mid p \in \mathrm{BPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = RA(f) \\ \{\epsilon\} \cup \{i.p \mid p \in \mathrm{BPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = DP(f) \\ \{\epsilon\} \cup \{i.p \mid p \in \mathrm{TPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = DCP(f) \\ \{\epsilon\} \cup \{i.p \mid p \in \mathrm{BPath}(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n), i = UCP(f) \\ \{\epsilon\} & \textit{otherwise} \end{cases}$$

Clearly, $\mathrm{TPath}(t)$ *and* $\mathrm{BPath}(t)$ *are totally ordered w.r.t. $\leq$ and the greatest element in* $\mathrm{TPath}(t)$ *and* $\mathrm{BPath}(t)$ *are called the maximum top path and the maximum bottom path and denoted by $TP(t)$ and $BP(t)$, respectively.*

DEFINITION 18.*[head/tail parts[20, 21]] Let t be a monomorphic term. For each $p \in$* $\mathrm{TPath}(t)$*, its head context $Ctop_{t,p}$ as well as for each $q \in$* $\mathrm{BPath}(t)$*, its tail context $Cbot_{t,q}$ are defined as follows:*

$$Ctop_{t,p} \;=\; \begin{cases} \Box & \textit{if } p = \epsilon \\ t[Ctop_{t_i,p'}]_i & \textit{if } p = i.p',\, t \equiv f(t_1,\ldots,t_n), \textit{ and } i = RA(f) \\ Ctop_{t_i,p'} & \textit{if } p = i.p',\, t \equiv f(t_1,\ldots,t_n), \textit{ and } i = UP(f) \\ t[Cbot_{t_i,q'}]_i & \textit{if } p = i.q',\, t \equiv f(t_1,\ldots,t_n), \textit{ and } i = DCP(f) \\ t[Ctop_{t_i,p'}]_i & \textit{if } p = i.p',\, t \equiv f(t_1,\ldots,t_n), \textit{ and } i = UCP(f) \end{cases}$$

$$Cbot_{t,q} \;=\; \begin{cases} \Box & \textit{if } q = \epsilon \\ Cbot_{t_i,q'} & \textit{if } q = i.q',\, t \equiv f(t_1,\ldots,t_n) \textit{ and } i = RA(f) \\ Cbot_{t_i,q'} & \textit{if } q = i.q',\, t \equiv f(t_1,\ldots,t_n) \textit{ and } i = DP(f) \\ t[Ctop_{t_i,p'}]_i & \textit{if } q = i.p',\, t \equiv f(t_1,\ldots,t_n) \textit{ and } i = DCP(f) \\ t[Cbot_{t_i,q'}]_i & \textit{if } q = i.q',\, t \equiv f(t_1,\ldots,t_n) \textit{ and } i = UCP(f) \end{cases}$$

The head part is given by $top(t,p) \equiv Ctop_{t,p}[t/p]$ and the tail part is by $bot(t,q) \equiv Cbot_{t,q}[t/q]$.

The development of the term patition based on prominent paths is solely based on the characterization lemmas for five types of argument positions. Thus this term partition can be corrected and extended based on our revised definition of monomorphic signature and argument positions given in the previous section. We refer to [20, 21] the definition of $ntp(t,p)$ and $nbt(t,q)$ in the following proposition.

PROPOSITION 19.*[term partition via prominent path (Theorem 36 of [21])] Let $\mathcal{R}$ be a sufficiently complete, terminating, and confluent TRS. Let t be a monomorphic term. (1) For* *each top path p in t, $\langle top(t,p), ntp(t,p)\rangle$ is a term partition of t. (2) For each bottom path q in t, $\langle nbt(t,q), bot(t,q)\rangle$ is a term partition of t.*

The sound generalization is obtained from **Proposition 19**.

PROPOSITION 20.*[sound generalization theorem (Theorem 37 of [21])] Let $\mathcal{R}$ be a sufficiently complete, terminating, and confluent TRS. Let $s \doteq t$ be a monomorphic equation and x be a fresh variable.*

1. *Let p be a top path in s and q a top path in t. Suppose that $s/p \equiv t/q$ and $top(s,p) \equiv top(t,q)$. Then $\mathcal{R} \vdash_{ind} s \doteq t$ iff $\mathcal{R} \vdash_{ind} s[x]_p \doteq t[x]_q$.*
2. *Let p be a bottom path in s and q a bottom path in t. Suppose that $s/p \equiv t/q$ and $bot(s,p) \equiv bot(t,q)$. Then $\mathcal{R} \vdash_{ind} s \doteq t$ iff $\mathcal{R} \vdash_{ind} s[x]_p \doteq t[x]_q$.*

5 Conclusion

We presented an example showing that the sound generalization proposed in [21] does not work without a condition imposed in their previous paper [20] that proposes induction based on term partition. We restored a condition in the definition of one of the argument positions and gave the corrected proof of the characterization of the position. Based on this, the correctness of sound generalization [21] was recovered. We note that all examples of sound genereralization presented in [21] still works under the restored condition. We also extended the technique by eliminating one of the restriction of monomorphic signature, localizing a part of the conditions for target term rewriting systems, and extending the notion of contextual positions. The corrected part of sound generalization is implemented in our experimental induction prover based on rewriting induction [1].

Despite the relaxation, some strong restrictions of monomorphicness are still imposed on the sound generalization. Finding other types of sound generalization applicable for non-monomorphic equations remains as a future work. Another future work is obtaining a lemma discovery method other than term partition and sound generalization via deeper analysis of monomorphicness.

Acknowledgments

Thanks are due to anonymous referees for helpful comments. This work was partially supported by a grant from JSPS, No. 20500002.

References

[1] T. Aoto. Designing a rewriting induction prover with an increased capability of non-orientable equations. In *Proc. of Symbolic Computation in Software Science Austrian-Japanese Workshop*, volume 08-08 of *RISC Technical Report*, pages 1–15, 2008.

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

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

[4] R. Boyer and J. Moore. *A Computational Logic*. Academic Press, 1979.

[5] A. Bundy. The automation of proof by mathmatical induction. In *Handbook of Automated Reasoning*, volume 1, pages 845–908. MIT Press, 2001.

[6] A. Bundy, D. Basin, D. Hutter, and A. Ireland. *Rippling: Meta-Level Guidance for Mathematical Reasoning*. Cambridge University Press, 2005.

[7] H. Comon. Inductionless induction. In *Handbook of Automated Reasoning*, volume 1, pages 913–962. MIT Press, 2001.

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

[9] G. Huet and J.-M. Hullot. Proof by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25(2):239–266, 1982.

[10] G. Huet and D. C. Oppen. Equations and rewrite rules: a survey. Technical report, Stanford University, Stanford, CA, USA, 1980.

[11] D. Hutter and C. Sengler. INKA: The next generation. In *Proc. of the 13th International Conference on Automated Deduction*, pages 288–292, 1996.

[12] A. Ireland and A. Bundy. Productive use of failure in inductive proof. *Journal of Automated Reasoning*, 16(1–2):79–111, 1996.

[13] D. Kapur, J. Giesl, and M. Subramaniam. Induction and decision procedures. *Revista de la real academia de ciencas (RACSAM) Serie A: Matematicas*, 98(1):154–180, 2004.

[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. Lemma discovery in automating induction. In *Proc. of the 13th International Conference on Automated Deduction*, volume 1104 of *LNCS*, pages 538–552. Springer-Verlag, 1996.

[16] D. Kapur and H. Zhang. An overview of rewrite rule laboratory (RRL). *Journal of Computer Mathematics with Applications*, 29(2):91–114, 1995.

[17] M. Kaufmann, P. Manolios, and J. S. Moore. *Computer-Aided Reasoning: ACL2 Case Studies*. Kluwer Academic Publishers, 2000.

[18] S. Shimazu, T. Aoto, and Y. Toyama. Automated lemma generation for rewriting induction with disproof. In *Proc. of the 8th JSSST Workshop on Programming and Programming Languages*, pages 75–89, 2006. In Japanese.

[19] S. Stratulat. A general framework to build contextual cover set induction provers. *Journal of Symbolic Computation*, 32:403–445, 2001.

[20] P. Urso and E. Kounalis. Term partition for mathematical induction. In *Proc. of the 14th International Conference on Rewriting Techniques and Applications*, volume 2706 of *LNCS*, pages 352–366, 2003.

[21] P. Urso and E. Kounalis. Sound generalizations in mathematical induction. *Theoretical Computer Science*, 323:443–471, 2004.

[22] T. Walsh. A divergence critic for inductive proof. *Journal of Artificial Intelligence Research*, 4:209–235, 1996.