Lemma Generation Method in Rewriting Induction for Constrained Term Rewriting Systems

Naoki Nakabayashi Naoki Nishida Keiichirou Kusakari
Toshiki Sakabe Masahiko Sakai

Recently, rewriting induction, which is one of the induction principles for proving inductive theorems in equational theory, has been extended to deal with constrained term rewriting systems. Rewriting induction has been applied to developing a method for proving the equivalence of imperative programs. To prove inductive theorems, there are many cases where appropriate lemmas need to be added. To this end, several methods for lemma generation in term rewriting have been studied. However, these existing methods are not effective for cases in constrained term rewriting. In this paper, we propose a framework of lemma generation for constrained term rewriting systems, in which we formalize the correspondences of terms in diverging equations by means of given constrained rewrite rules. We show an instance of the formalization, and also show that due to the framework with the instance, there is no necessity to give lemmas in advance for the examples shown by previous works.

1 Introduction

Program verification methods for imperative programs and functional programs are being researched using different approaches. The representative methods for verifying imperative programs are verification methods based on model checking [11] [26] [18] or on Hoare logic [15] [14] [18]. However, heuristic tasks are often necessary, such as providing checking algorithms that satisfy the specifications in model checking, as well as discovering loop invariant expressions and providing pre-conditions and post-conditions in Hoare logic. These heuristic tasks cannot, at times, be automated in such a way that they invariably provide the expected information. Conversely, Isabelle/Hol [24], AOL2 [21], PVS [13], Coq [12], SPIKE [6] [7] [29] and so on are representative theorem provers for functional programs. In term rewriting systems (TRS), which are one kind of computation model for functional programs, proof methods based on the principles of implicit induction, such as inductionless induction and rewriting induction, are being widely studied [23] [16] [27] [30] [8] [36] [1] [2] [3] [9] as automated verification methods for inductive theorems. Automated proving methods for inductive theorems can be used in equivalence verification because the equivalence of functions can be formulated as inductive theorems in term rewriting systems in such a way that they terminate with no adverse effects. However, there are times when, depending on the input, the verification procedure enters an endless loop and the verification fails.

A verification method that reduces the equivalence of imperative programs operating on natural numbers to inductive theorems in constrained term rewriting systems (constrained TRS), including the comparison of natural numbers in their constraints, has been proposed [41] based on inductionless in-

Graduate School of Information Science, Nagoya University

E-mail: nishida@is.nagoya-u.ac.jp (N. Nishida)

This work has been partially supported by *MEXT KAKENHI* #18500011, #20300010, #20500008 and #21700011, and the *Kayamori Foundation of Informational Science Advancement*.

This is a translated version of the article appeared in *Computer Software* (Vol. 28, No. 1, pp. 173–189, February 2010).

duction. This method is based on a completion procedure in constrained term rewriting systems, and it is possible to prevent endless looping of the verification procedure in a TRS using a comparison operator for natural numbers. Meanwhile, a new problem has arisen of automating the decomposition processing of the constrained part of the equation. It is also inherently difficult to handle integers under the restrictions of the completion procedure. And so, a verification method has been proposed in the literature [37] to verify by rewriting induction of inductive theorems in constrained term rewriting systems. With this method, it is no longer necessary to decompose the constraints and integers can be easily handled. This, however, presumes that the lemma equations needed for the proof have already been provided.

Generally speaking, it often necessary to provide appropriate lemma equations for automated proofs of inductive theorems. One way to obtain candidate lemma equations is to generalize (partially replace into variables) diverging equations. Although appropriate lemmas can be generated simply by generalizing maximal subterms with multiple appearances, erroneous lemmas are often generated depending on the timing of lemma generation during the theorem proof. Methods for generating more appropriate lemmas in unconstrained term rewriting systems have been proposed in the past, such as divergence critic [32], sound generalization [1] [31] [39], and the papers [10] [19] [20]. It may be possible to extend these methods to constrained TRSs, but the proofs in the papers [37] [41] show that these methods are not effective for generating the necessary lemmas. For example, the divergence critic is not effective for the divergence in the examples in the papers [37] [41]. Also sound generalization cannot be applied to rewriting rules that enable all of the arguments on the left side to be variables as the examples in [37] [41] show. Section 7 provides a more detailed comparison.

This paper formulates term relations from the rewriting rules and proposes a framework to generalize constrained equations based on those relations. In particular, constrained equations are generalized using the following procedure.

1. Generate rewriting rules that show term relations from the rewriting rules. At this point, a special constant that expresses the parts that are not to be generalized is introduced.
2. Extract the subterms in the equation parts and list them.
3. In the rules generated in step 1, those items matching the term at the top of the list are taken to be generalized terms. For the second and following terms, if a term that generalized the term one ahead is rewritten by rules generated in step 1 and the obtained item can be matched, that term is taken to be a generalized term. Each of the terms in the equation parts are generalized, complying with the generalized terms.
4. By generalizing the constraint, the constrained equation is generalized in accordance with both of the sides originally held by the equation and with the equivalent relations of the subterms in the constraint.

Also, rules of inference for adding lemmas are included among the rules of inference for the writing induction proposed in the paper [37], and the proofs are shown to be correct by means of the rewriting induction constructed from all those rules of inference. Finally, we include functions to generate, and add lemma equations in a proving procedure based on rewriting induction; we propose one generation method for rules that express term relations from the rewriting rules, and we show successful proofs for the exemplary proofs shown in [37] by automatic generation with this method without providing lemma equations in advance.

This paper has the following organization. Section 2 describes the rules for abstract reduction systems, terms, and constraints. Section 3 describes constrained term rewriting systems. Section 4 proposes a generalization of constrained equations. Section 5 proposes a rewriting induction procedure, in addition to the method proposed in Section 4 and proves its correctness. Section 5 also shows examples of proofs with this method. Section 6 compares this research with related research. Finally, Section 7 takes a look at future issues.

2 Preliminaries

This paper follows the general notation for term rewriting systems [4].

An *abstract reduction system* S is a pair $(A, \rightarrow)$, where the the *reduction* $\rightarrow$ is a binary relation on the set A. The reflexive transitive closure of $\rightarrow$ is

denoted by $\rightarrow^*$. An element $a \in A$ is called a *normal form* of S if there is no element $b \in A$ such that $a \rightarrow b$. An element $c \in A$ is said to *have a normal form w.r.t.* S if there exists a normal form a such that $c \rightarrow^* a$. The set of normal forms with respect to S is denoted by NF_S. S is called *Church-Rosser* (CR) if for each $x, y \in A$ with $x \leftrightarrow^* y$, there exists an element $z \in A$ such that $x \rightarrow^* z \leftarrow^* y$. S is said to be *terminating* if there exists no infinite reduction sequence of $\rightarrow$ starting from any element $a \in A$. The relation $\rightarrow^!$ is defined as follows:

$$\rightarrow^! = \{(a, b) \mid a \rightarrow^* b \in NF_S\}$$

The set of *terms* over a set $\mathcal{F}$ of *function symbols* (*signature*) and the countably infinite set $\mathcal{V}$ of *variables* is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. A term is called *ground* if it contains no variable, and the set of ground terms is denoted by $\mathcal{T}(\mathcal{F})$. For a term t, the set of variables appearing in t is denoted by $\mathcal{V}ar(t)$. We write $s \equiv t$ if terms s and t are identical. The set of *positions* in term t is denoted by $\mathcal{P}os(t)$:

- $\mathcal{P}os(t) = \{\varepsilon\}$ if $t \in \mathcal{V}$, and
- $\mathcal{P}os(t) = \{\varepsilon\} \cup \{iu \mid 1 \leq i \leq n,\ u \in \mathcal{P}os(t_i)\}$ if $t \equiv f(t_1, \ldots, t_n)$

where ε denotes the empty sequence. Note that the *root* position is represented as ε. The *hole* $\Box \notin \mathcal{F}$ is taken to be a special constant symbol. A *context* is a term in $\mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$ that includes exactly one $\Box$. The hole itself is also a context, and this kind of context is called an empty context. For a context $C[\ ]_p$ with the hole at position p, the term obtained by replacing the hole with a term t is denoted by $C[t]_p$. We may omit p from $C[\ ]_p$ and $C[t]_p$ if p is clear from the context. The set of contexts over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}_\Box(\mathcal{F}, \mathcal{V})$. For a term t, a term u is called a *subterm* of t, denoted by $t|_p$, if there exists a context $C[\ ]_p$ such that $t \equiv C[u]_p$. For a variable x appearing in a term t at position p, the occurrence of x at position p is called *shallow* if the length of p is at most one.†1 The *size* of a term is the total number of occurrences of function symbols and variables in the term.

For a *substitution* σ, thee *domain* and *range* of σ are denoted by $\mathcal{D}om(\sigma)$ $(= \{x \mid x \not\equiv \sigma(x)\})$ and $\mathcal{R}an(\sigma)$ $(= \{\sigma(x) \mid x \in \mathcal{D}om(\sigma)\})$, respectively, where the domain is finite. We may write $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ instead of σ if $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\}$ and $\sigma(x_i) \equiv t_i$. The application $\sigma(t)$ of σ to term t is called an *instance of* t and we may abbreviate it to $t\sigma$. We say that t is *more general than* $t\sigma$. For substitutions σ and σ', we write $\sigma = \sigma'$ if $\mathcal{D}om(\sigma) = \mathcal{D}om(\sigma')$ and $x\sigma \equiv x\sigma'$ for all variables x (in $\mathcal{D}om(\sigma)$). The composition $\sigma\sigma\prime$ of σ and σ' is defined as $x(\sigma\sigma') \equiv (x\sigma)\sigma'$. For a set X of variables, the restriction $\sigma|_X$ of σ to X is defined defined as $\{x \mapsto x\sigma \mid x \in \mathcal{D}om(\sigma) \cap X\}$. For substitutions σ and θ, we write $\sigma \leq \theta$ if there exists a substitution δ such that $\sigma\delta = \theta$. We say that σ is *ground to term* t if $t\sigma$ is ground. The substitution σ is simply called *ground* if σ is ground to any term appearing after σ. Given an abstract reduction system $S = (\mathcal{T}(\mathcal{F}, \mathcal{V}), \rightarrow)$, a ground substitution σ is called *normalized w.r.t.* S if $\mathcal{R}an(\sigma) \subseteq NF_S$.

Terms s and t are called *unifiable* if there exits a substitution σ such that $s\sigma \equiv t\sigma$. Then, σ is said to be a *unifier* of s and t. A unifier σ of s and t is called *most general* (*mgu*) if σ is more general than any unifier of s and t. Note that a most general unifier is unique up to variable renaming. We denote a most general unifier of s and t by $mgu(s, t)$.

A partial order $\succ$ over terms is called a *reduction order* if $\succ$ is well-founded and closed under contexts and substitutions.

Let $\mathcal{G}$ be a signature and $\mathcal{P}$ be a set of *predicate symbols*. Formulas over $\mathcal{G}$, $\mathcal{P}$ and $\mathcal{V}$ are defined by BNF notation as follows:

$$\phi ::= P(t_1, \ldots, t_n) \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \Rightarrow \phi \mid \phi \Leftrightarrow \phi \mid \top \mid \bot$$

where $P \in \mathcal{P}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{G}, \mathcal{V})$. $P(t_1, \ldots, t_n)$ is called an *atomic formula*. We assume that the binary equality predicate symbol EQ is included in $\mathcal{P}$. Note that this paper deals with quantifier-free formulas only. The set of *free variables* in formula ϕ is denoted by $fv(\phi)$.†2 The set of formulas over $\mathcal{G}, \mathcal{P}$ is denoted by $\mathcal{F}ol(\mathcal{G}, \mathcal{P}, \mathcal{V})$. A formula is called *closed* if it contains any free variable. The application of substitution σ to formula ϕ, denoted by $\phi\sigma$, is defined as the replacement of free variables with the corresponding terms registered in σ, and, in applying σ to ϕ, we implicitly assume that $\mathcal{R}an(\sigma|_{fv(\phi)}) \subseteq \mathcal{T}(\mathcal{G}, \mathcal{V})$. Positions of formulas are defined as well as those of terms by considering formulas as terms.

†1 In other words, p is either ε or a natural number.

†2 All the variables in quantifier-free formulas are free variables.

A *structure* $\mathcal{M}$ for $\mathcal{G}$ and $\mathcal{P}$ consists of the non-empty set A, called the *universe*, and the interpretation for $\mathcal{G}$ and $\mathcal{P}$. Every n-ary function symbol $g \in \mathcal{G}$ is interpreted by an n-ary function $g^{\mathcal{M}} : A^n \mapsto A$. Every n-ary predicate symbol $P \in \mathcal{P}$ is interpreted by an n-ary function $P^{\mathcal{M}} : A^n \mapsto \{\top, \bot\}$, where $\top$ and $\bot$ correspond to *true* and *false*, respectively. The interpretation of EQ is done by the function $EQ^{\mathcal{M}}$ that returns $\top$ if the value of the arguments are equivalent, and otherwise returns $\bot$.

Given a structure $\mathcal{M}$ for $\mathcal{G}$ and $\mathcal{P}$, we call a formula over $\mathcal{G}$, $\mathcal{P}$ and $\mathcal{V}$ a *constraint* (over $\mathcal{M}$).

The interpretation of ground terms (in $\mathcal{T}(\mathcal{G})$) w.r.t. $\mathcal{M}$ is defined as $(f(t_1, \ldots, t_n))^{\mathcal{M}} = f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}})$, and the interpretation of a closed atomic formula is defined as $(P(t_1, \ldots, t_n))^{\mathcal{M}} = P^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}})$. The interpretations of $\neg, \wedge, \vee, \top, \bot$ are defined as usual. We say that a closed formula ϕ *holds w.r.t.* $\mathcal{M}$, written as $\mathcal{M} \models \phi$, if $\phi^{\mathcal{M}} = \top$, and otherwise it *does not hold w.r.t.* $\mathcal{M}$, written as $\mathcal{M} \not\models \phi$. In this paper, we assume that the truth values of closed formulas are decidable.

A formula ϕ is called *valid w.r.t.* $\mathcal{M}$ if $\mathcal{M} \models \phi\sigma$ for every substitution σ such that $fv(\phi) \subseteq \mathcal{D}om(\sigma)$ and $\mathcal{R}an(\sigma|_{fv(\phi)}) \subseteq \mathcal{T}(\mathcal{G})$. We say that ϕ is called *satisfiable w.r.t.* $\mathcal{M}$ if there exists a substitution σ such that $fv(\phi) \subseteq \mathcal{D}om(\sigma)$, $\mathcal{R}an(\sigma|_{fv(\phi)}) \subseteq \mathcal{T}(\mathcal{G})$, and $\mathcal{M} \models \phi\sigma$. Furthermore, ϕ is called *unsatisfiable w.r.t.* $\mathcal{M}$ if ϕ is not satisfiable w.r.t. $\mathcal{M}$.

Example 2.1 Let $\mathcal{G}_{PA} = \{\mathsf{0}, \mathsf{s}, \mathsf{p}, +\}$, $\mathcal{P}_{PA} = \{=, \neq, <, \leq, >, \geq\}$, and $\mathcal{M}_{PA}$ be a structure for $\mathcal{G}_{PA}$ and $\mathcal{P}_{PA}$ such that the universe of $\mathcal{M}_{PA}$ is the set of integers, $\mathsf{0}^{\mathcal{M}_{PA}} = 0$, $\mathsf{s}^{\mathcal{M}_{PA}}(x) = x + 1$, $\mathsf{p}^{\mathcal{M}_{PA}}(x) = x - 1$, and the interpretations of the predicate symbols are the usual ones over integer. Here, $\mathcal{P}_{PA}$ does not include EQ, but we consider $=$ as EQ since $=^{\mathcal{M}_{PA}}$ coincides with the interpretation of EQ. Note that constraints over $\mathcal{M}_{PA}$ correspond to formulas in *Presburger arithmetic*

A closed formula in Presburger arithmetic is called p *Presburger sentence*. It is well known [25] [35] [40] that Presburger sentences[†3] is decidable. For this reason, the validity, satisfiability, and unsatisfiability for any constraint over $\mathcal{M}_{PA}$ is decidable.

3 Constrained Term Rewriting Systems and their Properties

This section introduces the definition of constrained term rewriting systems, which has been shown in the paper [37], and describes the characteristics that should be satisfied regarding constraints.

Let $\mathcal{F}$ and $\mathcal{G}$ be sets of function symbols consisting of $\mathcal{F} \cap \mathcal{G} = \emptyset$, let $\mathcal{P}$ be a set of predicate symbols, and let $\mathcal{M}$ be a structure for $\mathcal{G}$ and $\mathcal{P}$. Then, a *constrained rewriting rule over* $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$ is a triple (l, r, ϕ), written as $l \to r \ [\![\phi]\!]$, of the *left-hand side* l, the *right-hand side* r, and the *constrained part* ϕ, such that $l \notin \mathcal{V}$, $l, r \in \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{V})$, and $\mathcal{V}ar(l) \supseteq \mathcal{V}ar(r)$, $\phi \in \mathcal{F}ol(\mathcal{G}, \mathcal{P})$, and $\mathcal{V}ar(l) \supseteq fv(\phi)$. When ϕ is $\top$, the constraint may be omitted and it may be written as $l \to r$.

Let R be a finite set of constrained rewriting rules over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$. The *rewrite relation* $\to_R$ of R is defined as $\{(C[l\sigma]_p, C[r\sigma]_p) \mid l \to r \ [\![\phi]\!] \in R,\ C[\ ] \in \mathcal{T}_{\Box}(\mathcal{F} \cup \mathcal{G}, \mathcal{V}),\ \mathcal{M} \models \phi\sigma\}$.[†4] If the rewrite position p is to be made clear, we write $\to_{p,R}$ instead of $\to_R$. A *constrained term rewriting system over* $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$ is an abstract reduction system $(\mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{V}), \to_R)$ determined by the set $\mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{V})$ of terms and the rewrite relation $\to_R$, and is simply denoted by the set R of rewriting rules. When the constrained parts of all the rules in R are $\top$, R is a *term rewriting system* (*TRS*).

A *constrained equation over* $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$ is a triple (s, t, ϕ), written as $s \approx t \ [\![\phi]\!]$, of terms s, t and the *constrained part* ϕ, such that $s, t \in \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{V})$. When ϕ is $\top$, the constrained part may be omitted and the equation may be written as $s \approx t$. Also, $s \simeq t \ [\![\phi]\!]$ expresses $s \approx t \ [\![\phi]\!]$ or $t \approx s \ [\![\phi]\!]$, e.g., $s \simeq t \ [\![\phi]\!] \in E$ means that $s \approx t \ [\![\phi]\!] \in E$ or $t \approx s \ [\![\phi]\!] \in E$. Hereinafter, when the meaning is clear enough, constrained rewriting rules (equations) are simply called *rewriting rules* (*equations*). Each rewriting rule (equation) may have a unique label and be expressed as $\rho : l \to r \ [\![\phi]\!]$ ($\rho' : s \approx t \ [\![\psi]\!]$). Also, $s \approx t$ is called the *equation*

†3 In this paper, we assumed that constraints are quantifier-free. However, when constraints we treat are in Presburger arithmetic, we can allow constraints to contain quantifiers while the discussion later also holds for such constraints. This is because Presburger sentences are decidable even if they contain quantifiers.

†4 At this point, $\mathcal{R}an(\sigma|_{fv(\phi)}) \subseteq \mathcal{T}(\mathcal{G})$.

part of $s \approx t\ [\![\psi]\!]$.

Let E be a finite set of constrained equations over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$. The binary relation $\leftrightarrow_E$ determined by E is defined as $\{(C[s\sigma_g], C[t\sigma_g]) \mid s \simeq t\ [\![\phi]\!] \in E,\ C[\,] \in \mathcal{T}_\Box(\mathcal{F}, \mathcal{V}),\ \mathcal{M} \models \phi\sigma\}$.

A constrained equation $s \approx t\ [\![\phi]\!]$ is called an *inductive theorem of* R if $\mathcal{M} \models \phi\sigma_g$ and $s\sigma_g \leftrightarrow_R^* t\sigma_g$ for any arbitrary ground substitution σ_g such that $\mathcal{V}ar(s,t) \cup fv(\phi) \subseteq \mathcal{D}om(\sigma_g)$ and $\mathcal{R}an(\sigma_g|_{\mathcal{V}ar(s,t)\cup fv(\phi)}) \subseteq \mathcal{T}(\mathcal{F} \cup \mathcal{G})$ [9][37]. Given a term t and a constraint ϕ, a position p of t is called an *R-complete occurrence in t under ϕ* if $t\sigma_{NF}$ can be rewritten at position p for any ground normalized substitution σ_{NF} such that $\mathcal{M} \models \phi\sigma_{NF}$.

Next, we introduce the characteristics requested of constrained TRSs that uses rewriting induction.

Definition 3.1 ([37]) *Let R be a constrained term rewriting system over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$.*

- *R is called* complete w.r.t. *$\mathcal{M}$ if $\mathcal{M} \models EQ(s_g, t_g)$ for any arbitrary ground terms $s_g, t_g \in \mathcal{T}(\mathcal{G})$ such that $s_g \leftrightarrow_R^* t_g$.*
- *R is called* locally sound w.r.t. *$\mathcal{M}$ if, for any ground term $s_g \in \mathcal{T}(\mathcal{G})$, $t_g \in \mathcal{T}(\mathcal{G})$ and $\mathcal{M} \models EQ(s_g, t_g)$ for any ground term $t \in \mathcal{T}(\mathcal{F} \cup \mathcal{G})$ such that $s_g \to_R t_g$.*

Completeness ensures that if two terms are semantically equivalent, then they are also equivalent w.r.t. the rewrite relation. Local soundness ensures that a term obtained by rewriting an interpretable term has the same value of the original term.

Theorem 3.2 ([37]) *Let R be a constrained term rewriting system over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$.*

- *R is complete w.r.t. $\mathcal{M}$ if all of the following hold:*
  - *R is locally sound w.r.t. $\mathcal{M}$,*
  - *R is terminating, and*
  - *for any ground terms $s_g, t_g \in \mathcal{T}(\mathcal{G})$, if $s_g, t_g \in NF_R$ and $\mathcal{M} \models EQ(s_g, t_g)$, then $s_g \equiv t_g$.*
- *R is locally sound w.r.t. $\mathcal{M}$ if and only if, for any rewriting rule $l \to r\ [\![\phi]\!] \in R$, if $l \in \mathcal{T}(\mathcal{G}, \mathcal{V})$, then $r \in \mathcal{T}(\mathcal{G}, \mathcal{V})$ and $\phi \Rightarrow EQ(l, r)$ is valid w.r.t. $\mathcal{M}$.*

Example 3.3 ([37]) Let us consider the following constrained term rewriting system over $(\emptyset, \mathcal{G}_{PA}, \mathcal{P}_{PA}, \mathcal{M}_{PA})$ such that constrained parts correspond to formulas for Presburger arithmetics shown in $\mathcal{G}_{PA}, \mathcal{P}_{PA}, \mathcal{M}_{PA}$ of Example 2.1:

$$R_{\mathsf{add}} = \left\{ \begin{array}{c} \mathsf{0} + y \to y \\ \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\ \mathsf{p}(x) + y \to \mathsf{p}(x + y) \end{array} \right\}$$

R_{add} is locally sound w.r.t. $\mathcal{M}_{PA}$ since all the left-hand sides and the right-hand sides are included in $\mathcal{T}(\mathcal{G}_{PA}, \mathcal{V})$, and because $\mathsf{0} + y = y$, $\mathsf{s}(x) + y = \mathsf{s}(x + y)$, $\mathsf{p}(x) + y = \mathsf{p}(x + y)$ all are valid w.r.t. $\mathcal{M}_{PA}$. Also, because R_{add} has the CR property, R is sound w.r.t. $\mathcal{M}_{PA}$. However, when we consider $\mathsf{0}$ and $\mathsf{s}(\mathsf{p}(\mathsf{0}))$, although $\mathcal{M}_{PA} \models \mathsf{0} = \mathsf{s}(\mathsf{p}(\mathsf{0}))$, R_{add} does not satisfy $\mathsf{0} \leftrightarrow_{R_{\mathsf{add}}}^* \mathsf{s}(\mathsf{p}(\mathsf{0}))$. For this reason, R_{add} is not complete w.r.t. $\mathcal{M}_{PA}$.

Next we consider the following constrained term rewriting system over $(\emptyset, \mathcal{G}_{PA}, \mathcal{P}_{PA}, \mathcal{M}_{PA})$:

$$R_{PA} = R_{\mathsf{add}} \cup \left\{ \begin{array}{c} \mathsf{s}(\mathsf{p}(x)) \to x \\ \mathsf{p}(\mathsf{s}(x)) \to x \end{array} \right\}$$

R_{PA} is terminating and locally sound w.r.t. $\mathcal{M}_{PA}$. Moreover, if $s_g, t_g \in NF_R$ and $\mathcal{M}_{PA} \models s_g = t_g$, then $s_g \equiv t_g$ can be said. For this reason, R_{PA} is complete w.r.t. $\mathcal{M}_{PA}$.

4 Generalization of Constrained Equations

This section describes how lemma equations are generated, and defines properties and functions that are necessary for the generation.

Let us consider the following constrained TRS over $(\{\mathsf{sum}, \mathsf{sum1}, \mathsf{u}\}, \mathcal{G}_{PA}, \mathcal{P}_{PA}, \mathcal{M}_{PA})$:

$$R_{\mathsf{sum}} = R_{PA} \cup \left\{ \begin{array}{l} \mathsf{sum}(x) \to \mathsf{0}\ [\![x \le \mathsf{0}]\!] \\ \mathsf{sum}(\mathsf{s}(x)) \to \mathsf{sum}(x) + \mathsf{s}(x)\ [\![x \ge \mathsf{0}]\!] \\ \mathsf{sum1}(n) \to \mathsf{u}(n, \mathsf{s}(\mathsf{0}), \mathsf{0}) \\ \mathsf{u}(n, i, z) \to \mathsf{u}(n, \mathsf{s}(i), z + i)\ [\![i \le n]\!] \\ \mathsf{u}(n, i, z) \to z\ [\![i > n]\!] \end{array} \right\}$$

When proving that $\mathsf{sum}(n) \approx \mathsf{sum1}(n)$ is a inductive theorem of R_{sum} by using the method in [37], the generation of equations to be proved diverges as illustrated in Fig. 1. For the proof to succeed, it is sufficient to show that the first equation in Fig. 1 is an inductive theorem of R_{sum}. To this end, it suffices to prove that the second equation is

$$
\begin{array}{rcll}
\mathsf{u}(n,\mathsf{s}(0),0)+\mathsf{s}(n) & \approx & \mathsf{u}(\mathsf{s}(n),\mathsf{s}^2(0),0+\mathsf{s}(0)) & [\![\mathsf{s}(0)\le\mathsf{s}(n)]\!] \\
\mathsf{u}(n,\mathsf{s}^2(0),0+\mathsf{s}(0))+\mathsf{s}(n) & \approx & \mathsf{u}(\mathsf{s}(n),\mathsf{s}^3(0),(0+\mathsf{s}(0))+\mathsf{s}^2(0)) & [\![\mathsf{s}^2(0)\le\mathsf{s}(n)]\!] \\
\mathsf{u}(n,\mathsf{s}^3(0),(0+\mathsf{s}(0)+\mathsf{s}^2(0))+\mathsf{s}(n) & \approx & \mathsf{u}(\mathsf{s}(n),\mathsf{s}^4(0),((0+\mathsf{s}(0))+\mathsf{s}^2(0))+\mathsf{s}^3(0)) & [\![\mathsf{s}^3(0)\le\mathsf{s}(n)]\!] \\
 & \vdots & &
\end{array}
$$

Fig. 1 Example of divergence in the proof $\mathsf{sum}(n)\approx\mathsf{sum1}(n)$

an inductive theorem of R_{sum}, and to this end, we prove that the third equation is an inductive theorem of R_{sum}. This kind of repetition is a divergence of equations. If we generate an equation that is more general than some of equations in this divergence and if we succeed in proving that the more general equation is an inductive theorem, the all the original equations in the divergence are proved to be inductive theorems. As a result, divergence converges, and an overall proof can be completed.

Generally speaking, the cause of divergence in a proof is frequently the case that appropriate rewriting rules corresponding to induction hypotheses which are already obtained cannot be applied to equations to be proved. There is a possibility that the appropriate rewriting rules can be applied to more general equations, and thus, this technique can be expected to lead to a successful proof. For this reason, we try to generate a candidate of lemma equations from the second equation $\mathsf{u}(n,\mathsf{s}^2(0),0+\mathsf{s}(0))+\mathsf{s}(n)\approx\mathsf{u}(\mathsf{s}(n),\mathsf{s}^3(0),(0+\mathsf{s}(0))+\mathsf{s}^2(0))$ $[\![\mathsf{s}^2(0)\le\mathsf{s}(n)]\!]$ in Fig. 1. To prove that this equation is an inductive theorem, making $\mathsf{u}(n,x,y)+\mathsf{s}(n)\approx\mathsf{u}(\mathsf{s}(n),\mathsf{s}(x),y+x)$ $[\![x\le\mathsf{s}(n)]\!]$ a candidate of lemma equations enables us to use it as an induction hypothesis in later steps in the proof, and we can succeed in proving that the initial equation is an inductive theorem.

Next, we describe the idea of generating the candidate equation mentioned above. $\mathsf{u}(n,\mathsf{s}^2(0),0+\mathsf{s}(0))$ and $\mathsf{u}(\mathsf{s}(n),\mathsf{s}^3(0),(0+\mathsf{s}(0))+\mathsf{s}^2(0))$ are generalized into $\mathsf{u}(n,x,y)$ and $\mathsf{u}(\mathsf{s}(n),\mathsf{s}(x),y+x)$, respectively. This is caused from the fact that variations of the second and third arguments of u are the same. This correspondence coincides with the variation of the second and third arguments of u in the fourth rule of R_{sum}. On the other hand, we want to ignore the first argument of u since the first argument of u is not made a target of generalization. Viewed in this light, by using the constant Ω representing an arbitrary term, we deduce the following relation from the fourth rule in R_{sum}:

$$\rho_\Omega:\ \mathsf{u}(\Omega,i,z)\to\mathsf{u}(\Omega,\mathsf{s}(i),z+i)$$

In Section 5. 3, we will show a concrete method for generating ρ^Ω from R.

We denote by $\ge_\Omega$ the relation ensuring that Ω is more generalized than any arbitrary term. Then, the following relation holds between the terms mentioned above:

$$
\begin{array}{ccc}
\mathsf{u}(n,\mathsf{s}^2(0),0+\mathsf{s}(0)) & & \mathsf{u}(\mathsf{s}(n),\mathsf{s}^3(0),(0+\mathsf{s}(0))+\mathsf{s}^2(0)) \\
\curlyvee_\theta & & \curlyvee_\theta \\
\mathsf{u}(n,x,y) & & \mathsf{u}(\mathsf{s}(n),\mathsf{s}(x),y+x) \\
\vee|_\Omega & & \vee|_\Omega \\
\mathsf{u}(\Omega,x,y) & \to_{\varepsilon,\rho_\Omega} & \mathsf{u}(\Omega,\mathsf{s}(x),y+x)
\end{array}
$$

where $\theta=\{x\mapsto\mathsf{s}^2(0),\ y\mapsto 0+\mathsf{s}(0)\}$ and $t\prec_\theta s$ denotes the relation between s,t such that $s\equiv t\theta$.

On the other hand, constraint $\mathsf{s}^2(0)\le\mathsf{s}(n)$ must be generalized into $x\le\mathsf{s}(n)$. When the equation being focused on was generated, information can be obtained that $\mathsf{s}^2(0)$ in the constraint and the subterm $\mathsf{s}^2(0)$ in the second argument $\mathsf{s}^3(0)$ $(\equiv\mathsf{s}(\mathsf{s}^2(0)))$ of the right-hand side $\mathsf{u}(\mathsf{s}(n),\mathsf{s}^3(0),(0+\mathsf{s}(0))+\mathsf{s}^2(0))$ are equivalent (see Section 5. 1). Using this information, we obtain $x\le\mathsf{s}(n)$ along with the variable replacement mentioned in generalizing the equation part.

In the following, by following the above idea, we formalize a generalization method for constrained equations in a constrained TRS R over $(\mathcal{F},\mathcal{G},\mathcal{P},\mathcal{M})$.

First we provide definitions for Ω and $\ge_\Omega$.

Definition 4.1 ([22]) *Let $\Omega\notin\mathcal{F}$ be a fresh constant, and define the relation $\ge_\Omega$ over $\mathcal{T}(\mathcal{F}\cup\{\Omega\},\mathcal{V})$ as follows:*

- *$t\ge_\Omega\Omega$ and $t\ge_\Omega t$ for any term t, and*
- *$f(t_1,\ldots,t_n)\ \ge_\Omega\ f(s_1,\ldots,s_n)$ if $t_1\ \ge_\Omega\ s_1,\ \ldots,\ t_n\ge_\Omega s_n$.*

Example 4.2 $\mathsf{u}(n,x,y) \geq_\Omega \mathsf{u}(\Omega,x,y)$, $\mathsf{u}(\mathsf{s}(n),\mathsf{s}(x),y+x) \geq_\Omega \mathsf{u}(\Omega,\mathsf{s}(x),y+x)$.

Next we give a generalization method of equations.

Definition 4.3 *An equation $s \approx t\ [\![\phi]\!]$ is called an* instance *of an equation $s' \simeq t'\ [\![\phi']\!]$, and moreover, $s' \approx t'\ [\![\phi']\!]$ is said to be* more general than *$s \simeq t\ [\![\phi]\!]$ if there exists a substitution θ such that*

- *$s \equiv s'\theta$,*
- *$t \equiv t'\theta$, and*
- *$\phi \Leftrightarrow \phi'\theta$ is valid w.r.t. $\mathcal{M}$.*†5

Example 4.4 $\mathsf{u}(n,\mathsf{s}^2(0),0 + \mathsf{s}(0)) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n),\mathsf{s}^3(0),(0 + \mathsf{s}(0)) + \mathsf{s}^2(0))\ [\![\mathsf{s}^2(0) \leq \mathsf{s}(n)]\!]$ is an instance of $\mathsf{u}(n,x,y) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n),\mathsf{s}(x),y + x)\ [\![x \leq \mathsf{s}(n)]\!]$.

Let $s \approx t\ [\![\phi]\!]$ be an instance of $s' \simeq t'\ [\![\phi']\!]$. Then, it is clear by definition that if $s' \approx t'\ [\![\phi']\!]$ is an inductive theorem of R, then so is $s \approx t\ [\![\phi]\!]$.

Next, using a tuple of terms and rewriting rules R^Ω over $(\mathcal{F} \cup \{\Omega\},\mathcal{G},\mathcal{P},\mathcal{M})$, we define a function for generalizing the tuple of the terms. Let us write a tuple of n terms $s_1,\ldots,s_n$ as $(s_1,\ldots,s_n)$. Also, we will describe in Section 5. 3 some heuristics for generating R^Ω from R.

Definition 4.5 *Let R^Ω be a TRS over $(\mathcal{F} \cup \{\Omega\},\mathcal{G},\mathcal{P},\mathcal{M})$, and let $s_1,\ldots,s_n$ be terms in $\mathcal{T}(\mathcal{F} \cup \mathcal{G},\mathcal{V})$. We define the function genTerms as follows (see also Fig. 2):*

$$genTerms((s_1,\ldots,s_n),R^\Omega) = \left\{ (t_1,\ldots,t_n) \;\middle|\; \begin{array}{l} \exists\theta.\exists v_1,\ldots,v_n \in \mathcal{T}(\mathcal{F} \cup \mathcal{G} \cup \{\Omega\},\mathcal{V}). \\ \mathcal{D}om(\theta) \subseteq \mathcal{V}ar(v_1,\ldots,v_n), \\ (\forall i \in \{1,\ldots,n\}.\ s_i \equiv t_i\theta,\ v_i \leq_\Omega t_i), \\ (\forall i \in \{1,\ldots,n-1\}.\ v_i \to_{\varepsilon,R^\Omega} v_{i+1}) \end{array} \right\}$$

where v_1 is a variant of a left-hand side in R_Ω, and $t_1,\ldots,t_n$ are terms in $\mathcal{T}(\mathcal{F} \cup \mathcal{G},\mathcal{V})$ such that all of the following hold for all $1 \leq i \leq n$:

- *if $\forall p \in \mathcal{P}os(v_i).\ v_i|_p \equiv \Omega$, then $t_i|_p \equiv s_i|_p$, and*
- *t_i is most general in the terms satisfying the previous condition.*

Proposition 4.6 *genTerms is computable, and the returned set is finite up to variable renaming.*

†5 At this point, $\mathcal{R}an(\theta|_{fv(\phi')}) \subseteq \mathcal{T}(\mathcal{G},\mathcal{V})$ and $fv(\phi) = fv(\phi'\theta)$.

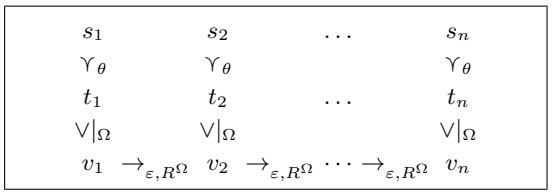

Fig. 2 Relations between s_i,t_i,v_i in the definition of *genTerms*.

Proof. The pair (t_1,v_1) of t_1 and v_1 belongs to the set $T_1 = \{(t,v) \mid \exists\theta.\ s \equiv t\theta \wedge t \geq_\Omega v \wedge (\exists\sigma.\ \exists l \to r \in R^\Omega.\ v \equiv l\sigma)\}$. The size of t is equal to or less than the size of s. If $t \geq_\Omega v$, then the size of v is equal to or less than the size of t. Therefore, T_1 is finite up to variable renaming. Let $V_{i+1} = \{v \mid v_i \to_{\varepsilon,R^\Omega} v\}$. Then, V_{i+1} is finite up to variable renaming since $\to_{\varepsilon,R^\Omega}$ is finitely branching. Since V_{i+1} is finite up to variable renaming , the set $T_{i+1} = \{(t,v) \mid \exists\theta.\ s_{i+1} \equiv t\theta \wedge t \geq_\Omega v \wedge v \in V_{i+1}\}$ is finite, and (t_{i+1},v_{i+1}) belongs to T_{i+1}. Due to the discussion above, *genTerms* is computable since the candidates of $(t_1,\ldots,t_n)$ are finite up to variable renaming. □

Example 4.7 For R_{sum}, we provide the following TRS R^Ω_{sum}:

$$R^\Omega_{\mathsf{sum}} = \left\{ \begin{array}{l} \mathsf{sum1}(\Omega) \to \mathsf{u}(\Omega,\mathsf{s}(0),0) \\ \mathsf{u}(\Omega,i,z) \to \mathsf{u}(\Omega,\mathsf{s}(i),z+i) \end{array} \right\}$$

Consider $s_1 \approx \mathsf{u}(n,\mathsf{s}^2(0),0 + \mathsf{s}(0))$, $s_2 \approx \mathsf{u}(\mathsf{s}(n),\mathsf{s}^3(0),(0 + \mathsf{s}(0)) + \mathsf{s}^2(0))$. Let $v_1 \equiv \mathsf{u}(\Omega,x,y)$, $v_2 \equiv \mathsf{u}(\Omega,\mathsf{s}(x),y + x)$, $\theta = \{x \mapsto \mathsf{s}^2(0),\ y \mapsto 0 + \mathsf{s}(0)\}$. Then, $genTerms((s_1,s_2),R^\Omega_{\mathsf{sum}}) = \{(\mathsf{u}(n,x,y),\mathsf{u}(\mathsf{s}(n),\mathsf{s}(x),y + x))\}$.

Next, we give a method for generalizing constraints.

Definition 4.8 *Let the labels expressing the left-hand side, right-hand side, and the constrained part for equations be* l, r, *and* c, *respectively. For an equation $s \approx t\ [\![\phi]\!]$, we use $\mathsf{l}.p$ to express the position p in the left-hand side s of the equation, use $\mathsf{r}.p$ to express the position p in the right-hand side t, and use $\mathsf{c}.p$ to express the position p in the con-*

straint ϕ:

$$(s \approx t\ [\![\phi]\!])|_{\mathsf{l}.p} \equiv s|_p$$
$$(s \approx t\ [\![\phi]\!])|_{\mathsf{r}.p} \equiv t|_p$$
$$(s \approx t\ [\![\phi]\!])|_{\mathsf{c}.p} \equiv \phi|_p$$

Let Q be a set of positions in equation e such that $e|_p \equiv e|_q$ for all $p, q \in Q$. Then, Q is said to be a position equivalence w.r.t. *e.*

We denote by $\mathcal{O}(s \approx t\ [\![\phi]\!])$ the set of all positions in the equation: $\mathcal{O}(s \approx t\ [\![\phi]\!]) = \{\mathsf{l}.p \mid p \in \mathcal{O}(s)\} \cup \{\mathsf{r}.p \mid p \in \mathcal{O}(t)\} \cup \{\mathsf{c}.p \mid p \in \mathcal{O}(\phi)\}$. In generating equations in proofs inferred by rewriting induction, we can compute position equivalences w.r.t. equations (see Subsection 5. 1). Note that we take the position equivalence of the initial set of equations to be empty. In this way, each equation ρ is assumed to have its own position equivalence. We denote the position equivalence by $\mathcal{O}_{=}(\rho)$.

Definition 4.9 *Suppose that the equation part of $\rho : s \approx t\ [\![\phi]\!]$ is generalized into $s' \approx t'$. Then, the function genCnst that generalizes ϕ is defined as follows:*

function *genCnst*$(\rho : s \approx t\ [\![\phi]\!], s', t') =$
  $\phi' := \phi$;
  $e' :== s' \approx t'$;
  for all $Q \in \mathcal{O}_{=}(\rho)$, the following is done:
    for all $\mathsf{c}.q \in Q$, the following is done:
      if $\mathsf{i}.p \in Q \wedge \mathsf{i} \in \{\mathsf{l}, \mathsf{r}\} \wedge \mathsf{i}.p \in \mathcal{O}(e')$
        $\wedge(\forall \mathsf{i}'.p' \in Q.\ \mathsf{i}'.p' \in \mathcal{O}(e'), e'|_{\mathsf{i}.p} \equiv e'|_{\mathsf{i}'.p'})$
      then $\phi' := \phi'[e'|_{\mathsf{i}.p}]_q$;
    end
  end
  return ϕ';

Example 4.10 Let ρ be $\mathsf{u}(n, \mathsf{s}^2(0), \mathsf{0} + \mathsf{s}(0)) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}^3(0), (\mathsf{0} + \mathsf{s}(0)) + \mathsf{s}^2(0))\ [\![\mathsf{s}^2(0) \leq \mathsf{s}(n)]\!]$ and $\mathcal{O}_{=}(\rho) = \{\{\mathsf{r}.2.1, \mathsf{r}.3.2, \mathsf{c}.1\}\}$. Then, $genCnst(\rho, \mathsf{u}(n, x, y) + \mathsf{s}(n), \mathsf{u}(\mathsf{s}(n), \mathsf{s}(x), y + x)) = x \leq \mathsf{s}(n)$.

Example 4.11 Let ρ be $\mathsf{f}(x, \mathsf{a}, \mathsf{a}) \approx \mathsf{g}(x)\ [\![\mathsf{p}(\mathsf{a})]\!]$ and $\mathcal{O}_{=}(\rho) = \{\{\mathsf{l}.2, \mathsf{l}.3, \mathsf{c}.1\}\}$. Then, $\mathsf{p}(\mathsf{a})$ is not generalized — $genCnst(\rho, \mathsf{f}(x, y, z), \mathsf{g}(x)) = \mathsf{p}(\mathsf{a})$ — since $(\mathsf{f}(x, y, z) \approx \mathsf{g}(x))|_{\mathsf{l}.2} \not\equiv (\mathsf{f}(x, y, z) \approx \mathsf{g}(x))|_{\mathsf{l}.3}$.

Finally, using *genTerms*, we define the function that generates generalized equations from equations and the rewriting rules R^{Ω}. If $i \neq j$ and t_i and t_j are not subterms of each other, then the notation $s[t_1/u_1, \ldots, t_n/u_n]$ uniquely represents the term obtained from s by replacing each t_i by u_i.

Definition 4.12 *Let $v_1, \ldots, v_n$ be subterms of the equation $s \approx t\ [\![\phi]\!]$ and not be subterms of each other. Then, we define the function genEqn that generates equations obtained from $s \approx t\ [\![\phi]\!]$, $(v_1, \ldots, v_n)$, and R^{Ω} as shown in Fig. 3.*

Example 4.13 Let ρ be $\mathsf{u}(n, \mathsf{s}^2(0), \mathsf{0} + \mathsf{s}(0)) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}^3(0), (\mathsf{0} + \mathsf{s}(0)) + \mathsf{s}^2(0))\ [\![\mathsf{s}^2(0) \leq \mathsf{s}(n)]\!]$, $\mathcal{O}_{=}(\rho) = \{\{\mathsf{r}.2.1, \mathsf{r}.3.2, \mathsf{c}.1\}\}$, and (s_1, s_2) be $(\mathsf{u}(n, \mathsf{s}^2(0), \mathsf{0} + \mathsf{s}(0)), \mathsf{u}(\mathsf{s}(n), \mathsf{s}^3(0), (\mathsf{0} + \mathsf{s}(0))))$. Then, $genEqn(\rho, R^{\Omega}_{\mathsf{sum}}, (s_1, s_2)) = \{\mathsf{u}(n, x, y) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}(x), y + x) + \mathsf{s}(n)\ [\![x \leq \mathsf{s}(n)]\!]\}$.

The following theorem holds for *genEqn*.

Theorem 4.14 *Let R^{Ω} be a TRS, e be an equation, $v_1, \ldots, v_n$ be subterms on both sides of e such that $v_1, \ldots, v_n$ are not subterms of each other. Then, any equation $e' \in genEqn(e, R^{\Omega}, (v_1, \ldots, v_n))$ is more general than e.*

Proof. Let e be $s \approx t\ [\![\phi]\!]$, and e' be $s' \simeq t'\ [\![\phi']\!]$. Then, by the definitions of *genTerms* and *genCnst*, there exists a substitution θ such that $s \equiv s'\theta$ and $t \equiv t'\theta$ and $\phi \equiv \phi'\theta$. Thus, e' is more general than e. □

The assumption that $v_1, \ldots, v_n$ are not subterms of each other ensures that the uniqueness of the replacement of subterms generated by *genTerms*.

Finally, we provide an intuitive description of the method above that is using R^{Ω}. This method bridges the disparity in the number of applications of the rewriting rules by applying R^{Ω}. For example, because of the constraint $\mathsf{s}^3(0) \leq \mathsf{s}(n)$ for the $\mathsf{s}(n)$ in the first argument, the term s_2 ($\equiv \mathsf{u}(\mathsf{s}(n), \mathsf{s}^3(0), (\mathsf{0} + \mathsf{s}(0)) + \mathsf{s}^2(0))$) in Example 4.7 has the rewriting rule $\mathsf{u}(n, i, z) \to \mathsf{u}(n, \mathsf{s}(i), z + i)\ [\![i \leq n]\!]$ applied once more than for the term s_1 ($\equiv \mathsf{u}(n, \mathsf{s}^2(0), \mathsf{0} + \mathsf{s}(0))$). For this reason, by applying to s_1 the rewriting rule $\mathsf{u}(\Omega, i, z) \to \mathsf{u}(\Omega, \mathsf{s}(i), z + i)$ in $R^{\Omega}_{\mathsf{sum}}$ corresponding to $\mathsf{u}(n, i, z) \to \mathsf{u}(n, \mathsf{s}(i), z + i)\ [\![i \leq n]\!]$, the disparity in the number of applications of the rule to s_1 and

$$genEqn(s \approx t\ [\![\phi]\!], R^{\Omega}, (v_1, \ldots, v_n)) = \left\{ s' \approx t'\ [\![\phi']\!] \;\middle|\; \begin{array}{l} (v'_1, \ldots, v'_n) \in genTerms((v_1, \ldots, v_n), R^{\Omega}),\ s' \equiv s[v_1/v'_1, \ldots, v_n/v'_n], \\ t' \equiv t[v_1/v'_1, \ldots, v_n/v'_n],\ \phi' = genCnst(s \approx t\ [\![\phi]\!], s', t') \end{array} \right\}$$

Fig. 3 Definition of *genEqn*

s_2 is bridged, and the equation is generalized.

5 Rewriting Induction in Constrained Term Rewriting Systems

This section introduces a rewriting induction method [37] for validating inductive theorems of constrained term rewriting systems, and expands the method so that lemma equations can be added. Then, we propose a strategy for automatically validating inductive theorems of constrained term rewriting systems.

A proof by rewriting induction is done by applying inference rules to a pair of a set E of equations to be proved, and a set H of rewriting rules that can be used in the proof. The rules in H can be seen as the induction hypotheses or lemmas that can be used in the proof of E. The proof starts from $(E, \emptyset)$, and if it can be transformed into $(\emptyset, H)$, the the proof is successfully completed.

5.1 Inference Rules

The inference rules of rewriting induction for constrained term rewriting systems are illustrated in Fig. 4 [37]. The function *Expd* used in Expansion is defined as follows.

Definition 5.1 *The function Expd is defined as follows:*

$$Expd(s \approx t\ [\![\phi]\!], p) = \left\{ \rho : C[r]_p\sigma \approx t\sigma\ [\![\&(\phi\sigma, \psi\sigma)]\!] \;\middle|\; \begin{array}{l} s \equiv C[u]_p, \\ l \to r\ [\![\psi]\!] \in R, \\ \sigma = mgu(u, l) \end{array} \right\}$$

where $l \to r\ [\![\psi]\!]$ and $s \approx t\ [\![\phi]\!]$ have no shared variable, and & is defined as follows:†6

$$\&(\phi\sigma, \psi\sigma) = \begin{cases} \psi\sigma & \text{if } \psi\sigma \Rightarrow \phi\sigma \text{ is valid w.r.t. } \mathcal{M} \text{ and } fv(\psi\sigma) = fv(\phi\sigma) \\ \phi\sigma & \text{if } \phi\sigma \Rightarrow \psi\sigma \text{ is valid w.r.t. } \mathcal{M} \text{ and } fv(\psi\sigma) = fv(\phi\sigma) \\ \phi\sigma \wedge \psi\sigma & \text{otherwise} \end{cases}$$

Moreover, when $\&(\phi\sigma, \psi\sigma) = \phi\sigma \wedge \psi\sigma$, the position equivalence for an equation ρ is defined as follows:

$$\mathcal{O}_{=}(\rho) = \bigcup_{x \in \mathcal{V}ar(r)} \{\{\mathsf{l}.p.q \mid r|_q \equiv x\} \cup \{\mathsf{c}.2.q \mid \psi|_q \equiv x\}\} \cup \bigcup_{x \in \mathcal{V}ar(t)} \{\{\mathsf{r}.q \mid t|_q \equiv x\} \cup \{\mathsf{c}.1.q \mid \phi|_q \equiv x\}\}$$

When $\&(\phi\sigma, \psi\sigma) = \phi\sigma$ or $\psi\sigma$, $\mathcal{O}_{=}(\rho)$ is defined by replacing $\mathsf{c}.2.q$ and $\mathsf{c}.1.q$ with $\mathsf{c}.q$. Furthermore, we remove from $\mathcal{O}_{=}(\rho)$ a set A satisfying one of the following conditions since such a relation is ineffective for generalizing constraints:

- *A does not contain any position in the form of $\mathsf{c}.p$,*
- *the number of elements is at most one, i.e., A is a singleton set, or*
- *$\rho|_p \in \mathcal{V}$ for $p \in A$.*†7

Furthermore, when $\phi\sigma \Leftrightarrow \psi\sigma$ is valid w.r.t. $\mathcal{M}$, whichever of $\phi\sigma$ or $\psi\sigma$ is larger than the set of $\mathcal{O}_{=}(\rho)$ is taken to be the result of $\&(\phi\sigma, \psi\sigma)$.

When the inference rules in Fig. 4 are applied once to (E, H), resulting in (E', H'), we write $(E, H) \vdash_{\mathrm{RI}} (E', H')$. The reflexive and transitive closure of $\vdash_{\mathrm{RI}}$ is denoted by $\vdash^{*}_{\mathrm{RI}}$. $\vdash^{\mathsf{e}}_{\mathrm{RI}}$ represents the application of Expansion. $E \uplus E'$ is identical to $E \cup E'$ and $E \cap E' = \emptyset$.

Next, we propose an inference rule of for soundly adding lemma equations into equation sets.†8 The inference rule is illustrated in Fig. 5. Postulate in Fig. 5 is based on the one in [27], and the rule is improved so that the disproving faculty in [37]. The

†6 Although the paper [37] defines & as $\&(\phi\sigma, \psi\sigma) = \phi\sigma \wedge \psi\sigma$, this paper modifies the definition to optimize constraints in order to make the generalization of constraints more concise.

†7 Note that $\rho|_{p_1} \equiv \cdots \equiv \rho|_{p_n}$ for $A = \{p_1, \ldots, p_n\}$.

†8 A lemma generation method is called *sound* if all the equations generated by the method from inductive theorems are inductive theorems.

Simplification

$$\frac{(E \uplus \{C[l\sigma] \simeq t \ [\![\phi]\!]\}, H)}{(E \cup \{C[r\sigma] \approx t \ [\![\phi]\!]\}, H)} \quad \text{if } l \to r \ [\![\psi]\!] \in R \cup H,\ \phi \text{ are satisfiable w.r.t. } \mathcal{M},\ fv(\psi\sigma) \subseteq fv(\phi), \text{ and } \phi \Rightarrow \psi\sigma \text{ is valid w.r.t. } \mathcal{M}$$

Deletion

$$\frac{(E \uplus \{s \simeq t \Leftarrow \phi\}, H)}{(E, H)} \quad \text{if } s \equiv t \text{ or } \phi \text{ is unsatisfiable w.r.t. } \mathcal{M}$$

Expansion

$$\frac{(E \uplus \{s \simeq t \Leftarrow \phi\}, H)}{(E \cup \mathit{Expd}(s \approx t \ [\![\phi]\!], p), H \cup \{s \to t \Leftarrow \phi\})} \quad \text{if } s \succ t,\ s \notin \mathcal{V},\ \mathcal{V}ar(s) \supseteq \mathcal{V}ar(t),\ fv(\phi) \subseteq \mathcal{V}ar(s), \text{ and } p \text{ is an } R\text{-complete position of } s \text{ under } \phi$$

EQ-Deletion

$$\frac{(E \uplus \{C[s_1, \ldots, s_n] \simeq C[t_1, \ldots, t_n] \ [\![\phi]\!]\}, H)}{(E \cup \{C[s_1, \ldots, s_n] \approx C[t_1, \ldots, t_n] \ [\![\phi \wedge \neg(\bigwedge_{i=1}^{n} EQ(s_i, t_i))]\!]\}, H)} \quad \text{if } \{s_1, t_1, \ldots, s_n, t_n\} \subseteq \mathcal{T}(\mathcal{G}, \mathcal{V}) \text{ and } \mathcal{V}ar(s_i, t_i) \subseteq fv(\phi)$$

Fig. 4 Inference rules of rewriting induction for constrained term rewriting systems [37]

inference rule is essentially the same as the inference rule in [2] for adding sound lemma equations to equation sets. The difference from [2] is that E can include equations that are not inductive theorems. This is caused from the fact that the lemma generation method in this paper is not sound in general. Moreover, L is not added to E; instead, for the sake of efficiency in the implementation, H' obtained in the proof of L is added to H. Actually, we choose one of the equations obtained by generating candidates of lemma equations as the element of L (see Subsection 5. 2). As mentioned above, the lemma generation method in this paper is not sound. However, the generated lemma candidates are themselves first proved independently of the main proof, and their induction hypotheses generated in the proof are added to the main proof stream only if they are inductive theorems. Due to this mechanism (use of Postulate), we preserve the soundness of proofs. When the inference rules in Fig. 4 and Fig. 5 are applied once to (E, H), resulting in (E', H'), we write $(E, H) \vdash_{\mathrm{RI+Pos}} (E', H')$. The reflexive and transitive closure of $\vdash_{\mathrm{RI+Pos}}$ is denoted by $\vdash^*_{\mathrm{RI+Pos}}$. $\vdash_{\mathrm{Pos}}$ represents the application of Postulate. $\vdash^*_{\mathrm{RI+Pos}}$ is used for $\vdash^*$ in Fig. 5. $(E, H) \vdash^*_{\mathrm{RI+Pos}} (E', H')$ is called the *proof*.

The following lemma and theorem hold for the inference rules in Fig. 4 and Fig. 5.

Lemma 5.2 *If $(E, H) \vdash^*_{\mathrm{RI+Pos}} (E', H')$, there exists an equation set $E_0 \subseteq E_0$ such that $(E_0, H) \vdash^*_{\mathrm{RI}} (E', H')$ and every equation in $E_0 \setminus E$ is an inductive theorem of R.*

Postulate

$$\frac{(E, H)}{(E, H \cup H')} \quad \text{if } (L, \emptyset) \vdash^* (\emptyset, H')$$

Fig. 5 An inference rule for adding lemmas.

Proof. Suppose that $(E_i, H_i) \vdash_{\mathrm{Pos}} (E_{i+1}, H_{i+1})$. Then, $E_i = E_{i+1}$, and there exists a set L such that $(L, \emptyset) \vdash^*_{\mathrm{RI+Pos}} (\emptyset, H'_i)$ and $H_{i+1} = H_i \cup H'_i$. Moreover, every equation in L is an inductive theorem of R. Thus, the proof holds even if E_i, H_i is added to the proof: $(E_i \cup L, H_i) \vdash^*_{\mathrm{RI+Pos}} (E_i, H_i \cup H'_i) = (E_{i+1}, H_{i+1})$. By repeating this operation, we obtain a proof that does not use Postulate. Also, let E'_1 be the set of equations added by the application of Postulate. Then, $(E \cup E'_1, H) \vdash^*_{\mathrm{RI+Pos}} (E', H')$. Moreover, every equation in E'_1 is an inductive theorem of R. □

Theorem 5.3 *Let R be a constrained term rewriting system over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$. Let E be a finite set of constrained equations over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{M})$, and let $\succ$ be a reduction order such that $\to_R \subseteq \succ$. Suppose that R is complete and locally sound w.r.t. $\mathcal{M}$. If $(E, \emptyset) \vdash^*_{\mathrm{RI+Pos}} (\emptyset, H)$, then all the equations in E*

are inductive theorems of R.

Proof. This theorem is clear from the soundness of $\vdash^*_{\mathrm{RI}}$ [37] and Lemma 5.2. □

5.2 Strategy for Applying the Inference Rules

To implement automated proofs in a computer, the order of application of the inference rules must be fixed. This section improves the strategy in [37] for applying the inference rules for rewriting induction of constrained TRSs by adapting it to the generation of lemma equations.

The following considerations were obtained in experiments up to now:

(1) More proofs are successful when the induction hypotheses (rules H) are given application priority over the rewriting rules in R.

(2) When we apply basic rewriting rules such as addition (for example, the rule R_{PA}) to equations, there are cases where the generation method for candidates of lemma equations proposed in the previous section is ineffective. For example, when both sides of an equation introduced in Section 4 (the second equation in Fig. 1) are reduced by R_{PA} to normal forms, we obtain $\mathsf{u}(n, \mathsf{s}^2(\mathsf{0}), \mathsf{s}(\mathsf{0})) + \mathsf{s}(n) \approx \mathsf{u}(n, \mathsf{s}^3(\mathsf{0}), \mathsf{s}^4(\mathsf{0}))\ [\![\mathsf{s}^2(\mathsf{0}) \leq \mathsf{s}(n)]\!]$, and it is difficult to deduce the desired lemma equation by using the generation method.

To formalize the proof strategy for consideration (1) above, let us define a rewriting relation that takes into consideration the priority for a set of rewriting rules.

Definition 5.4 *We define the* rewriting relation $\to_{R[\phi]}$ under a satisfiable constraint ϕ *as follows:* $\to_{R[\phi]} = \{(C[l\sigma], C[r\sigma]) \mid l \to r\ [\![\psi]\!] \in R,\ C[\ ] \in \mathcal{T}_{\square}(\mathcal{F}, \mathcal{V}),\ fv(\psi\sigma) \subseteq fv(\phi),\ \phi \Rightarrow \psi\sigma$ *is valid w.r.t.* $\mathcal{M}\}$.

Definition 5.5 *Let $[R_1, \ldots, R_n]$ be a list of sets for rewriting rules. We define the rewriting relation $\to_{[R_1,\ldots,R_n][\phi]}$ that takes in consideration the priority of $[R_1, \ldots, R_n]$, as follows: $s \to_{[R_1,\ldots,R_n][\phi]} t$ if either of the following holds:*

- $s \to_{R_1[\phi]} t$, *or*
- *there exists no term t' with $s \to_{R_1[\phi]} t'$, and $s \to_{[R_2,\ldots,R_n][\phi]} t$.*

$\to_{[R_1,\ldots,R_n][\phi]}$ is slightly different from the reduction of *priority term rewriting system* [5] [28]; an order is given between the sets of rewriting rules. By applying Simplification with using $\to_{[H,R][\phi]}$, the rules H can be prioritized and applied.

Let R be a constrained TRS. Let $R_{\mathcal{G}}$ be the subset over $(\emptyset, \mathcal{G}, \mathcal{P}, \mathcal{M})$, and $R_{\mathcal{F}} = R \setminus R_{\mathcal{G}}$. We suppose that every equation in E has its own label attached. We denote the set of labels for E by $\mathrm{Label}(E)$: $\mathrm{Label}(E) = \{\rho \mid \rho : s \approx t\ [\![\phi]\!] \in E\}$.

Table 1 shows the definitions that apply to each of the inference rules multiple times: $\Rightarrow_{\mathrm{Simp}}$, $\Rightarrow_{\mathrm{EQ-Del}}$, and $\Rightarrow_{\mathrm{Del}}$ is the application of Simplification, EQ-Deletion, and Deletion multiple times, respectively. $\Rightarrow_{\mathrm{Simp}[R_1,\ldots,R_n]\&\mathrm{Del}}$ is an application of the inference rules, that adapts to the problem described in consideration (2) above, and is used concretely as $[R_1, \ldots, R_n] = [H, R_{\mathcal{F}}, R_{\mathcal{G}}]$. Equations that could not be deleted by Deletion in $\Rightarrow_{\mathrm{Simp}[R_1,\ldots,R_n]\&\mathrm{Del}}$ are restored to the equations before Simplification was applied. Also, we assume that, when both sides of an equation e are rewritten, the position equivalence $\mathcal{O}_{=}(e)$ of $e$is appropriately updated by following the calculation [17] of the descendant of positions.

Definition 5.6 *Let R be a constrained TRS, let $e = s \approx t\ [\![\phi]\!]$, and let $e' = s' \approx t'\ [\![\phi]\!]$ be an equation obtained from e by reducing either s or t once. Let the rewriting at this time be $C[u]_p \to_{p,R} C[u']_p$, and the applied rewriting rule be $l \to r\ [\![\psi]\!]$. Moreover, when the left side is rewritten, let $\mathsf{i} = \mathsf{l}$, and otherwise, $\mathsf{i} = \mathsf{r}$. Then, we define the relation between variable positions in l and the corresponding positions in r as follows:*

$$\eta = \{(\mathsf{i}.pq, \{\mathsf{i}.pq' \mid r|_{q'} \equiv l|_q\}) \mid l|_q \in \mathcal{V}\}$$

When consider η a mapping, the domain of η is denoted by $\mathcal{D}om(\eta)$: $\mathcal{D}om(\eta) = \{\mathsf{i}.pq \mid l|_p \in \mathcal{V}\}$. We define the function update_η *that replaces the prefix of the positions in accordance with η as follows:*

Table 1 the definitions of multiple application of inference rules.

Relation	Definition
$(E,H) \Rightarrow_{\mathrm{Simp}[R_1,\ldots,R_n]} (E',H')$	$E' = \{s' \approx t' \ [\![\phi]\!] \mid s \approx t \ [\![\phi]\!] \in E,\ s \to^{!}_{[R_1,\ldots,R_n][\phi]} s',\ t \to^{!}_{[R_1,\ldots,R_n][\phi]} t'\}$
$(E,H) \Rightarrow_{\mathrm{EQ-Del}} (E',H')$	Apply EQ-Deletion to each equation in E, to which EQ-Deletion can be applied, obtaining E'.
$(E,H) \Rightarrow_{\mathrm{Del}} (E',H')$	Apply Deletion as much as possible to E, obtaining E'.
$(E,H) \Rightarrow_{\mathrm{Simp}[R_1,\ldots,R_n]\&\mathrm{Del}} (E'',H')$	$(E,H) \Rightarrow_{\mathrm{Simp}[R_1,\ldots,R_n]} \Rightarrow_{\mathrm{EQ-Del}} \Rightarrow_{\mathrm{Del}} (E',H')$, $E'' = \{\rho \in E \mid \rho \in \mathit{Label}(E')\}$

$$\mathrm{update}_\eta(p) = \begin{cases} \{q''q \mid q'' \in P\} & \text{if } (p',P) \in \eta \text{ and } p = p'q \\ \{p\} & \text{otherwise} \end{cases}$$

$$\mathrm{update}_\eta(Q) = \bigcup_{\substack{p \in P \\ n}} \eta(p)$$

$$\mathrm{update}_\eta(\{Q_1,\ldots,Q_n\}) = \bigcup_{i=1} \{\mathrm{update}_\eta(Q_i)\}$$

Using update_η *and* $\mathcal{O}_=(e)$*, the position equivalence* $\mathcal{O}_=(e')$ *for* e' *is defined as* $\mathrm{update}_\eta(\mathcal{O}_=(e))$*.*

Let R be a constrained term rewriting system over $(\mathcal{F},\mathcal{G},\mathcal{P},\mathcal{M})$, E be a finite set of constrained equations over $(\mathcal{F},\mathcal{G},\mathcal{P},\mathcal{M})$, and let $\succ$ be a reduction order such that $\to_R \subseteq \succ$. Moreover, suppose that R is complete and locally sound w.r.t. $\mathcal{M}$. Then, we propose a *verification procedure for inductive theorems* that uses the inference rules in Fig. 4, 5 and in Fig. 6. Users can freely specify R^Ω and subterm, which makes tuples of subterms that are not subterms of each other. Section 5. 3 introduces a concrete example. If this verification procedure results in $(\emptyset,H)$, all of the equations included in E are inductive theorems of R. This is because from Theorem 5.3, regardless of the application of inference rules, the resulting proof is correct.

5. 3 Heuristics for Generating Lemmas

This paper provides the following heuristics for generating lemma equations.

- R^Ω is the set obtained from rewriting rules $l \to r \ [\![\phi]\!]$ in R such that both the root symbols of l and r belong to $\mathcal{F}$, by replacing variables appearing at *shallow* positions only by Ω and by removing the constrained part (Example 4.7). The variables that can be replaced in Ω correspond to arguments that are not changed by the rewriting.
- The sequence subterm(e) of subterms in equation e that is a target of lemma generation, is a sequence of maximal subterms whose root symbols belong to $\mathcal{F}$ and are the same as the root of a left-hand side in R^Ω, that is incremental in the sense of the size of terms.
- The prediction of whether the equation is diverging when an equation is added during an expansion, is determined by referencing the remembered rules in R in the history,[†9] and we consider an equation diverging if a remembered rule appears two or more times consecutively. In other words, Expansion by the same rule to an equation (and its descendants) is allowed up to twice, and if a third time seems likely, we attempt to generate a lemma. To simplify the description in the examples in this paper, this was limited to one or more times. However, even for two or more times the proof is successful in the same way.
- The candidates e' in 3. in Fig. 6 are restricted to ones whose constrained parts are generalized.

5. 4 Exemplary Proofs

This subsection describes examples of various proofs using this method. The proofs in this paper use the following strategy.

- Expansion uses termination of $R \cup H \cup \{s \to t \ [\![\phi]\!]\}$ instead of the reduction orders $\succ$ [34] [33].
- A leftmost outermost strategy is used in the rewriting for the $\to_{R_i[\phi]}$ in $[R_1,\ldots,R_n]$ in $\Rightarrow_{\mathrm{Simp}[R_1,\ldots,R_n]}$.

†9 When an equation is generated by *Expd*, $l \to r \ [\![\psi]\!]$ can be added to the history of $s \approx t \ [\![\phi]\!]$ and thus, can be remembered.

Let $H := \emptyset$. Then, we apply inference rules to (E, H) in the following way:

1. $(E, H) \Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}}]} (E', H')$, $E := E'$.
2. $(E, H) \Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}},R_{\mathcal{G}}]\&\mathrm{Del}} (E', H')$. If $E' = \emptyset$, then the procedure halts successfully. Otherwise, let $E := E'$ and go to 3.
3. For an equation e selected due to a heuristics and for which divergence is predicted, given $e' \in genEqn(e, R^{\Omega}, \mathrm{subterm}(e))$, if $(\{e'\}, \emptyset) \vdash^{*}_{\mathrm{RI}} (\emptyset, H')$, then let $H := H \cup H'$ and go to 1 (i.e., we apply Postulate). Otherwise, go to 4.
4. Apply Expansion once and go to 1.

Fig. 6 A verification procedure based on rewriting induction.

- When equations and so forth are selected from sets, we selected the one detected at the first time (i.e., the head element of lists representing sets). However, in the examples in this paper, the choice was always at most one.
- If there exists two or more R-complete positions in an equation, we choose the leftmost outermost one.

Also, all the proofs were made automatically with humans following the proof strategy of this paper.

Example 5.7 An example proving that $\mathsf{sum}(n) \approx \mathsf{sum1}(n)$ is an inductive theorem is illustrated in Fig. 7. As shown in Fig. 7, this proof was successful without providing lemma equations in advance.

Example 5.8 Let us consider whether $\mathsf{fib}(n) \approx \mathsf{fib1}(n)$ can be proved to be an inductive theorem of the following constrained TRS:

$$R_{\mathsf{fib}} = R_{PA} \cup \left\{ \begin{array}{r} \mathsf{fib}(x) \to 0 \ [\![x \le 0]\!] \\ \mathsf{fib}(\mathsf{s}(0)) \to \mathsf{s}(0) \\ \mathsf{fib}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{fib}(\mathsf{s}(x)) + \mathsf{fib}(x) \ [\![x \ge 0]\!] \\ \mathsf{fib1}(n) \to \mathsf{u1}(n, \mathsf{s}(0), 0, \mathsf{s}(0)) \\ \mathsf{u1}(n, i, y, z) \to \mathsf{u1}(n, \mathsf{s}(i), z, y + z) \ [\![i \le n]\!] \\ \mathsf{u1}(n, i, y, z) \to y \ [\![i < n]\!] \end{array} \right\}$$

When *genEqn* is applied to the equations in the diverging equation $\mathsf{u1}(\mathsf{s}(n), \mathsf{s}^3(0), 0 + \mathsf{s}(0), \mathsf{s}(0) + (0 + \mathsf{s}(0))) + \mathsf{u1}(n, \mathsf{s}^2(0), \mathsf{s}(0), 0 + \mathsf{s}(0)) \approx \mathsf{u1}(\mathsf{s}^2(n), \mathsf{s}^4(0), \mathsf{s}(0) + (0 + \mathsf{s}(0)), (0 + \mathsf{s}(0)) + (\mathsf{s}(0) + (0 + \mathsf{s}(0)))) \ [\![\mathsf{s}^3(0) \le \mathsf{s}^2(n)]\!]$, the candidate of lemma equations $\mathsf{u1}(\mathsf{s}(n), \mathsf{s}(x_1), x_3, x_2 + x_3) + \mathsf{u1}(n, x_1, x_2, x_3) \approx \mathsf{u1}(\mathsf{s}^2(n), \mathsf{s}^2(x_1), x_2 + x_3, x_3 + (x_2 + x_3)) \ [\![\mathsf{s}(x_1) \le \mathsf{s}^2(n)]\!]$ is generated and the proof is successful without providing any lemma equations in advance.

Example 5.9 Let us consider whether $\mathsf{fact}(n) \approx \mathsf{fact1}(n)$ can be proved to be an inductive theorem of the following constrained TRS:

$$R_{\mathsf{fact}} = R_{PA} \cup \left\{ \begin{array}{r} \mathsf{fact}(x) \to 0 \ [\![x \le 0]\!] \\ \mathsf{fact}(\mathsf{s}(x)) \to \mathsf{sum}(x) \times \mathsf{s}(x) \ [\![x \ge 0]\!] \\ \mathsf{fact1}(n) \to \mathsf{u2}(n, \mathsf{s}(0), 0) \\ \mathsf{u2}(n, i, z) \to \mathsf{u2}(n, \mathsf{s}(i), z \times i) \ [\![i \le n]\!] \\ \mathsf{u2}(n, i, z) \to z \ [\![i > n]\!] \\ 0 \times y \to 0 \\ \mathsf{s}(x) \times y \to x \times y + y \end{array} \right\}$$

This example closely resembles Example 5.7, but the point of difference is that the $\times$ appearing on the right-hand side of $\mathsf{u2}$ is a function symbol in $R_{\mathcal{F}}$. Then, as for the example of R_{sum}, when *genEqn* is applied to the equations in the diverging equation $\mathsf{u2}(n, \mathsf{s}^2(0), 0 + \mathsf{s}(0)) \times \mathsf{s}(n) \approx \mathsf{u2}(\mathsf{s}(n), \mathsf{s}^3(0), (0 + \mathsf{s}(0)) \times \mathsf{s}^2(0)) \ [\![\mathsf{s}^2(0) \le \mathsf{s}(n)]\!]$, the candidate of lemma equations $\mathsf{u2}(n, x_1, x_2) \times \mathsf{s}(n) \approx \mathsf{u2}(\mathsf{s}(n), \mathsf{s}(x_1), x_2 \times x_1) \ [\![x_1 \le \mathsf{s}(n)]\!]$ is generated, and the proof is successful without providing any lemma equations in advance.

Finally, we introduce an example that cannot generate lemmas for successful proofs in the framework (Fig. 2) for this method.

Example 5.10 Let us consider whether $\mathsf{double}(n) \approx \mathsf{double1}(n)$ can be proved to be an inductive theorem of the following constrained TRS:

$$R_{\mathsf{double}} = R_{PA} \cup \left\{ \begin{array}{r} \mathsf{double}(x) \to 0 \ [\![x \le 0]\!] \\ \mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x))) \ [\![x > 0]\!] \\ \mathsf{double1}(n) \to \mathsf{u3}(n, \mathsf{s}(0), 0) \\ \mathsf{u3}(n, i, z) \to \mathsf{u3}(n, \mathsf{s}(i), \mathsf{inc2}(z)) \ [\![i \le n]\!] \\ \mathsf{u3}(n, i, z) \to z \ [\![i > n]\!] \\ \mathsf{inc2}(n) \to \mathsf{s}(\mathsf{s}(n)) \end{array} \right\}$$

Then, even if *genEqn* is applied to the equa-

$$
\begin{array}{l}
\quad (\{\mathsf{sum}(n) \approx \mathsf{sum1}(n)\}, \emptyset) \\
\Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}}]} \\
\left(\left\{ \quad \mathsf{sum}(n) \approx \mathsf{u}(n, \mathsf{s}(0), 0) \quad \right\}, \emptyset \right) \\
\vdash^{\mathrm{e}}_{\mathrm{RI}} \\
\left(\left\{ \begin{array}{rl} & 0 \approx \mathsf{u}(n, \mathsf{s}(0), 0) \ [\![n \leq 0]\!] \\ (2) & \mathsf{sum}(n) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}(0), 0) \ [\![n \geq 0]\!] \end{array} \right\}, \left\{ \ (1) \quad \mathsf{sum}(n) \to \mathsf{u}(n, \mathsf{s}(0), 0) \ \right\} \right) \\
\Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}}]} \\
\left(\left\{ \begin{array}{rl} & 0 \approx 0 \ [\![n \leq 0]\!] \\ (2) & \mathsf{u}(n, \mathsf{s}(0), 0) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}^2(0), 0 + \mathsf{s}(0)) \ [\![n \geq 0]\!] \end{array} \right\}, \{(1)\} \right) \\
\Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}},R_{\mathcal{G}}]\&\mathrm{Del}} \\
\left(\ \{(2)\}, \{(1)\} \ \right) \\
\vdash^{\mathrm{e}}_{\mathrm{RI}} \\
\left(\begin{array}{l} \left\{ \begin{array}{rl} & \mathsf{u}(n, \mathsf{s}(0), 0) + \mathsf{s}(n) \approx 0 + \mathsf{s}(0) \ [\![n \geq 0 \wedge \mathsf{s}(0) > n]\!] \\ (3) & \mathsf{u}(n, \mathsf{s}(0), 0) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}^3(0), (0 + \mathsf{s}(0)) + \mathsf{s}^2(0)) \ [\![\mathsf{s}^2(0) \leq \mathsf{s}(n)]\!] \end{array} \right\}, \\ \left\{ \begin{array}{c} (1) \\ (2) \quad \mathsf{u}(\mathsf{s}(n), \mathsf{s}^2(0), 0 + \mathsf{s}(0)) \ \to \ \mathsf{u}(n, \mathsf{s}(0), 0) + \mathsf{s}(n) \ [\![n \geq 0]\!] \end{array} \right\} \end{array} \right) \\
\Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}}]} \hookrightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}},R_{\mathcal{G}}]\&\mathrm{Del}} \\
\left(\left\{ \ (3) \quad \mathsf{u}(n, \mathsf{s}^2(0), 0 + \mathsf{s}(0)) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}^3(0), (0 + \mathsf{s}(0)) + \mathsf{s}^2(0)) \ [\![\mathsf{s}^2(0) \leq \mathsf{s}(n)]\!] \ \right\}, \left\{ \begin{array}{c} (1) \\ (2) \end{array} \right\} \right)
\end{array}
$$

$$
\begin{array}{l}
\quad \text{By } \{\mathsf{u}(n,x,y) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}(x), y + x) \ [\![x \leq \mathsf{s}(n)]\!]\} = genEqn((3), R^{\Omega}_{\mathsf{sum}}, (\mathsf{u}(n, \mathsf{s}^2(0), 0 + \mathsf{s}(0)), \mathsf{u}(\mathsf{s}(n), \mathsf{s}^3(0), (0 + \mathsf{s}(0)) + \mathsf{s}^2(0))), \\
\left(\left\{ \quad \mathsf{u}(n,x,y) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}(x), y + x) \ [\![x \leq \mathsf{s}(n)]\!] \ \right\}, \left\{ \ \emptyset \ \right\} \right) \\
\vdash^{\mathrm{e}}_{\mathrm{RI}} \\
\left(\begin{array}{l} \left\{ \begin{array}{rl} & y + x \approx \mathsf{u}(n,x,y) + \mathsf{s}(n) \ [\![x \leq \mathsf{s}(n) \wedge \mathsf{s}(x) > \mathsf{s}(n)]\!] \\ & \mathsf{u}(n,x,y) + \mathsf{s}(n) \approx \mathsf{u}(\mathsf{s}(n), \mathsf{s}^2(x), (y + x) + \mathsf{s}(x)) \ [\![\mathsf{s}(x) \leq \mathsf{s}(n)]\!] \end{array} \right\}, \\ \left\{ \ (4) \quad \mathsf{u}(\mathsf{s}(n), \mathsf{s}(x), y + x) \to \mathsf{u}(n,x,y) + \mathsf{s}(n) \ [\![x \leq \mathsf{s}(n)]\!] \ \right\} \end{array} \right) \\
\Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}}]} \\
\left(\left\{ \begin{array}{rl} & y + x \approx \mathsf{u}(n,x,y) + \mathsf{s}(n) \ [\![\mathsf{s}(x) \leq \mathsf{s}(n) \wedge \mathsf{s}(x) > \mathsf{s}(n)]\!] \\ & \mathsf{u}(n, \mathsf{s}(x), y + x) + \mathsf{s}(n) \approx \mathsf{u}(n, \mathsf{s}(x), y + x) + \mathsf{s}(n) \ [\![\mathsf{s}(x) \leq \mathsf{s}(n)]\!] \end{array} \right\}, \left\{ \ (4) \ \right\} \right) \\
\Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}},R_{\mathcal{G}}]\&\mathrm{Del}} \\
\left(\ \emptyset, \ \{ \ (4) \ \} \ \right)
\end{array}
$$

$$
\begin{array}{l}
\vdash_{\mathrm{Pos}} \\
\left(\ (3), \ \{ \ (1), (2), (4) \ \} \ \right) \\
\Rightarrow_{\mathrm{Simp}[H,R_{\mathcal{F}},R_{\mathcal{G}}]\&\mathrm{Del}} \\
\left(\ \emptyset, \ \{ \ (1), (2), (4) \ \} \ \right)
\end{array}
$$

The position equivalences are the following: $\mathcal{O}_{=}((2)) = \emptyset$, $\mathcal{O}_{=}((3)) = \{\{\mathsf{r}.2.1, \mathsf{r}.3.2, \mathsf{c}.1\}\}$

Fig. 7 Exemplary proofs for $\mathsf{sum}(n) = \mathsf{sum1}(n)$

tions $\mathsf{s}^2(\mathsf{u3}(n, \mathsf{s}^3(0), \mathsf{s}^4(0))) \approx \mathsf{u3}(\mathsf{s}(n), \mathsf{s}^4(0), \mathsf{s}^6(0))$ in the diverging equation, the results are $t_2\theta = \mathsf{u3}(n, \mathsf{s}^3(0), \mathsf{inc2}(\mathsf{s}^4(0)))$, $s_2 = \mathsf{u3}(\mathsf{s}(n), \mathsf{s}^4(0), \mathsf{s}^6(0))$, and generalization is not possible. However, the relation $t_2\theta \to_R^* s_2$ works. Now, if the condition $t_i\theta \equiv s_i$ in Definition 4.5 is expanded to $t_i\theta \to_R^* s_i$, the candidate of lemma equations $\mathsf{s}^2(\mathsf{u3}(n, i, z)) \approx \mathsf{u3}(\mathsf{s}(n), \mathsf{s}(i), \mathsf{inc2}(z)) \ [\![i \leq \mathsf{s}(n)]\!]$ is generated, and the proof succeeds without providing a lemma equation in advance.

By testing this method on all of the examples mentioned in this section, we have confirmed that the method was successful in automatic proofs for all the examples. Example 5.10 was also expanded and, after the expansion, we succeed in proofs for all the examples.

6 Comparison with Related Research

This section compares this research with related research.

Divergence critic [32] is an effective method if one side of the equation diverges at its root. Because the function symbol u for R_{sum} in this paper has a rule that a symbol does not appear in u and rewriting cannot occur for both sides of the root, in the divergences occasioned by the u rules, the divergences do not occur on one side of the root of the equation. Thus, the divergence critic is not effective for the examples in this paper.

Sound generalization [31] is a method to obtain items such as the top path and bottom path from the form of the rewrite rules and to generate a sound lemma. Sound generalization is effective for a single-sorted TRS, and the instances introduced in this paper are single-sorted TRSs if their constraints are well thought out. However, because, in the rules for the function symbol u for R_{sum} in this paper, a pattern that is not a variable cannot appear in the argument on the left-hand side, the top path and bottom path cannot be obtained. For this reason, Sound generalization is not effective for the examples in this paper. Moreover, a method [1] that extends sound generalization and a method [39] that combines sound generalization and the divergence critic have both been proposed but these methods are not effective for the instances in this paper for the same reason. However, because a method that generates sound lemmas is very powerful, how to incorporate sound generalization remains an issue for the future.

The methods in the papers [20] [19] are effective lemma generating methods for function definitions with tail recursion that perform pattern matching with $\mathsf{0}$ or $\mathsf{s}(x)$ as the first argument on the left-hand side. A constrained TRS obtained from an imperative program by means of the transformations used in the papers [37] [41] has only variables in the first argument and it is a function definition that does case analysis on the constrained part. Thus, the methods in [20] [19] are not effective for the exemplary proofs in this paper.

In rippling [10], effective annotation cannot be done for rewriting rules that greatly change the structure of the terms, nor can they be generalized. Thus, it is not effective for exemplary proofs where generalization is intrinsically necessary, such as the examples in this paper.

Finally, we investigate whether this method can be applied to an unconstrained TRS. In a constrained TRS, further rewriting is possible in the normal form under the constraint by the R^{Ω} rule, which ignores the constraint. For that reason, lemma generation was successful. But because the normal form in an unconstrained TRS cannot be rewritten even once using R^{Ω} based on Section5. 3, R^{Ω} does not work. Thus, this method not effective for an unconstrained TRS. However, it is possible that this method could be extended so that it becomes an effective method even for an unconstrained TRS by applying ingenuity to the construction of R^{Ω} and to the application strategy for the rules of inference. This kind of extension remains an issue for the future.

7 Conclusion

This paper proposed a lemma equation candidate generation method for rewriting induction in constrained rewriting systems. It demonstrated that the correctness of the rewriting induction does not fail even if rules of inference for adding lemmas are added. Regarding the examples in this paper, proof was conclusively constructed based on this method. We implemented this method and confirmed that it was effective for the examples in this paper. We performed tests on various examples and analyzed the heuristics and strategy for proving they work. A discussion of the extension described in Example 5.10 remains an issue for the future.

During expansion, equations that cannot be oriented may appear and proving may fail because those equations cannot be oriented. The commutative law of addition is one typical example. In the paper [3], rewriting induction was extended to handle the kind of equations that cannot be oriented. The paper [37] supplements the commutative law of addition with EQ-Deletion and shows that proofs can be possible even if there are no equations corresponding to commutative laws. However, it is not always possible to compensate for equations that cannot be oriented with EQ-Deletion. Thus, by introducing the methods in the papers [2] [3] even into rewriting induction for constrained TRSs, we can expect an increase in provable examples. Expand-

ing this method to handle equations that cannot be oriented also remains an issue for the future.

In the simplification of this method, application of the rewrite rules H, which corresponds to an assumption in induction, was given priority over the rewrite rules R. This is a heuristic that was obtained when multiple instances were proved. The paper [38], however, points out by example that this strategy is not always effective and discusses this point. Thus, in a rewrite strategy for simplification, testing many instances to analyze what kind of strategy or what priority sequence for the rewriting rules is effective is necessary. This point, too, is an issue for our future research.

This method was successful in generating lemmas for the exemplary proofs; it transformed loops in imperative programs, for which current methods are ineffective in generating lemmas, and obtained a constrained TRS. Conversely, this method was not very effective for examples for which previous unconstrained methods successfully generated lemmas or for rewriting systems without loop invariant expressions. As another future issue, it is also necessary to investigate how to combine this method with previous methods.

References

[1] Aoto, T.:Sound lemma generation for proving inductive validity of equations, Proc. of the 28th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008), 2008, pp. 13–24, Leibniz International Proceedings in Informatics, Vol.2, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[2] Aoto, T.: Designing a Rewriting Induction Prover with an Increased Capability of Non-Orientable Theorems, *Proceedings of Austrian-Japanese Workshop on Symbolic Computation in Software Science* (2008), pp. 1–15

[3] Aoto, T.: Soundness of Rewriting Induction based on an Abstract Principle, *IPSJ Transactions on Programming*, Vol. 49, No. SIG 1 (PRO 35) (2008), pp. 28–38

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

[5] Baeten, J.C.M., Bergstra, J.A., Klop, J.W., and Weijland, W.P.: Term-Rewriting Systems with Rule Priorities, *Theoretical Computer Science*, Vol. 67, No. 2&3(1989), pp. 283–301.

[6] Bouhoula, A. and Rusinowitch,M.: Implicit Induction in Conditional Theories. *Journal of Automated Reasoning*, Vol. 14, No. 2 (1995), pp. 189–235.

[7] Bouhoula, A. and Rusinowitch,M.: SPIKE: A System for Automatic Inductive Proofs. *Proc. of the 4th International Conference on Algebraic Methodology and Software Technology*, Lecture Notes in Computer Science, Vol. 936 (1995), pp. 576–577.

[8] Bouhoula, A.: Automated Theorem Proving by Test Set Induction, *Journal of Symbolic Computation*, Vol. 23, No. 1(1997), pp. 47–77.

[9] Bouhoula, A. and Jacquemard, F.: Automated Induction with Constrained Tree Automata. *Proc. of the 4th International Joint Conference on Automated Reasoning*, Lecture Notes in Computer Science, Vol. 5195 (2008), Springer, pp. 539–554.

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

[11] Clarke, E. M. and Emerson, E. A.: Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic, *Proceedings of Logic and Programs Workshop*, Lecture Notes in Computer Science, Vol. 131(1981), pp. 52–71.

[12] Cornes, C., Courant, J., Filliatre, J. C., Huet, G., Murthy, C., Munoz, C., Parent, C., Symbolique, C., Manoury, P., Manoury, P., Saibi, A., Werner, B. and Projet Coq.: The Coq Proof Assistant - Reference Manual, 1995.

[13] Crow, J., Owre, S., Rushby, J., Shankar, N. and Srivas, M.: A Tutorial Introduction to PVS, 1995.

[14] Dijkstra, E. W.: A Discipline of Programming, Prentice-Hall, 1976.

[15] Hoare, C. A. R.: An Axiomatic Basis for Computer Programming, *Communications of the ACM*, Vol. 12, No. 10(1969), pp. 576–580.

[16] Huet, G. P. and Hullot, J.-M.: Proofs by Induction in Equational Theories with Constructors, *Journal of Computer and System Sciences*, Vol. 25, No. 2(1982), pp. 239–266.

[17] Huet, G. P. and Lévy, J.J.: Computations in Orthogonal Rewriting Systems, I and II, in: *Computational Logic, Essays in Honor of Alan Robinson*, The MIT Press, pp. 396–443.

[18] Huth, M. and Ryan, M.: *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2000.

[19] Kapur, D. and Sakhanenko, N.: Automatic Generation of Generalization Lemmas for Proving Properties of Tail-Recursive Definitions, *Proc. of the 16th International Conference on Theorem Proving in Higher Order Logics*, Lecture Notes in Computer Science, Vol. 2758, 2003, pp. 136–154.

[20] Kapur, D. and Sakhanenko, N.: Lemma Discovery in Automated Induction, *Proc. of the 13th International Conference on Automated Deduction*, Lecture Notes in Computer Science, Vol. 1104, 1996, pp. 538–552.

[21] Kaufmann, M. and Moore, J. S.: ACL2-Tutorial, 1997.

[22] Klop, J.W.: Term Rewriting systems, *Handbook of Logic in Computer Science*, Vol. 1, Oxford University Press, Oxford, 1992.

[23] Musser, D. R.: On Proving Inductive Properties of Abstract Data Types, *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages*, 1980, pp. 154–162.

[24] Nipkow, T., Paulson, L. C. and Wenzel, M.: Isabelle/HOL A Proof Assistant for Higher-Order Logic, Springer, 2008.

[25] Presburger, M.: Uber die Vollstandigkeit eines gewissen Systems der Arithmetic ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt, Compte-Rendus dei Congres des Mathematiciens des Pays Slavs, 1929.

[26] Queille, J.-P. and Sifakis, J.: Specification and Verification of Concurrent Systems in CESAR, *Proceedings of the 5th International Symposium on Programming*, Lecture Notes in Computer Science, Vol. 137(1982), Springer, pp. 337–351.

[27] Reddy, U. S.: Term Rewriting Induction, *Proceedings of the 10th International Conference on Automated Deduction*, Lecture Notes in Computer Science, Vol. 449(1990), Springer, pp. 162–177.

[28] Sakai, M. and Toyama, Y.: Semantics and Strong Sequentiality of Priority Term Rewriting Systems, *Theoretical Computer Science*, Vol. 208, 1998, pp. 87–110.

[29] Stratulat, S.: A General Framework to Build Contextual Cover Set Induction Provers. *Journal of Symbolic Computation*, Vol. 32, No. 4 (2001), pp. 403–445.

[30] Toyama, Y.: How to prove equivalence of term rewriting systems without induction, Theoretical Computer Science, Vol.90, pp.369-390, 1991.

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

[32] Walsh, T: A Divergence Critic for Inductive-Proof, *Journal of Artificial Intelligence Research*, Vol. 4(1996), pp. 209–235.

[33] Wehrman, I., Stump, A., and Westbrook, E.: Slothrop: Knuth-Bendix Completion with a Modern Termination Checker. *Proc. of the 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, Vol. 4098 (2006), Springer, pp. 287–296.

[34] Aoto, T.: Rewriting induction using termination checkers. *Proceedings of the 24th Conference of Japan Society for Software Science and Technology (JSSST)*, number 3C-2, pages 1–4, Sept. 2007 (in Japanese).

[35] Iri, M., Nozaki, A., and Noshita, K.: *KEISAN-no-Kouritsuka-to-sono-Genkai*. Nyumon Gendai-no-Sugaku [13], Nippon-Hyoron-sha Co. Ltd., 1980 (in Japanese).

[36] Koike, H. and Toyama, Y.: Inductionless induction and rewriting induction. *Computer Software*, 17(6):1–12, 2000 (in Japanese).

[37] Sakata, T., Nishida, N., Sakabe, T., Sakai, M., and Kusakari, K.: Rewriting induction for constrained term rewriting systems. *IPSJ Transactions on Programming*, 2(2):80–96, Mar. 2009 (in Japanese).

[38] Sato, H. and Kurihara, M.: Rewriting induction with multiple contexts. *Proceedings of the 26th Conference of Japan Society for Software Science and Technology (JSSST)*, number 7B-1, pages 1–13, Sept. 2009 (in Japanese).

[39] Shimazu, S., Aoto, T., and Toyama, Y.: Automated lemma generation for rewriting induction with disproof. *Computer Software*, 26(2):41–55, 2009 (in Japanese).

[40] Higashino, T., Kitamichi, J., and Taniguchi, K.: Presburger arithmetic and its application to program developments. *Computer Software*, 9(6):31–39, 1992 (in Japanese).

[41] Furuichi, Y., Nishida, N., Sakai, M., Kusakari, K., and Sakabe, T.: Approach to procedural-program verification based on implicit induction of constrained term rewriting systems. *IPSJ Transactions on Programming*, 1(2):100–121, Sept. 2008 (in Japanese).