# Logical Approach to Theorem Proving with Term Rewriting on KR-logic

Tadayuki Yoshida[1], Ekawit Nantajeewarawat[2], Masaharu Munetomo[3] and Kiyoshi Akama[3]

[1]*Tokyo Software Development Laboratory, International Business Machines Corporation, Tokyo, Japan*

[2]*Computer Science Program, Sirindhorn International Institute of Technology Thammasat University, Pathumthani, Thailand*

[3]*Information Initiative Center, Hokkaido University, Sapporo, Japan*

Keywords: Term Rewriting Rules, Logical Problem Solving Framework, Equivalent Transformation, Correctness.

Abstract: Term rewriting is often used for proving theorems. To mechanizing such a proof method with computation correctness guaranteed strictly, we follow LPSF, which is a general framework for generating logical problem solution methods. In place of the first-order logic, we use KR-logic, which has function variables, for correct formalization. By repeating (1) specialization by a substitution for usual variables, and (2) application of an already derived rewriting rule, we can generate a term rewriting rule from the resulting equational clause. The obtained term rewriting rules are proved to be equivalent transformation rules. The correctness of the computation results is guaranteed. This theory shows that LPSF integrates logical inference and functional rewriting under the broader concept of equivalent transformation.

## 1 INTRODUCTION

Term rewriting is often used for solving proof problems (Bird and Wadler, 1988; Dershowitz and Jouannaud, 1990). Typical examples are proofs in group theory, where the axiom of a group consists of laws, such as associativity law $\forall x,y,z \in G : x\cdot(y\cdot z) = (x\cdot y)\cdot z$. The operator $(\cdot)$ here is a mapping $G\times G \to G$. Assuming the axiom of a group, we want to prove that $((a^{-1}\cdot a)\cdot(b\cdot b^{-1}))^{-1} = b\cdot((a\cdot b)^{-1}\cdot a)$. Usually, term rewriting is used for proving such an equation. If the terms on both sides of this equation are rewritten repeatedly to reach a single term finally, the equation is proved.

Such a proof by term rewriting is, however, not regarded as logical problem solving. To mechanizing such a proof method, we need to formalize the problem as a logical problem with computation correctness guaranteed strictly. We follow the LPSF (Akama et al., 2019a), which is a general framework for generating logical problem solving methods for Model-Intersection (MI) problems (Akama and Nantajeewarawat, 2016). We apply the LPSF to this proof problem by reformalizing it as an MI problem.

We also need to construct a rule set. We already know how to make term rewriting rules from equational clauses (Akama et al., 2019b). Based on this work, we try to devise a new method of generating new term rewriting rules from *Cs*.

Assume that *Cs* is a background knowledge consisting of equational clauses. We apply to *Cs* repeatedly (1) specialization by a substitution for usual variables, and (2) application of already derived rewriting rules. After repetition, we make a term rewriting rule from the resulting equational clause. We prove the correctness of the obtained term rewriting rule, i.e., it is an ET rule. Since ET rules are repeatedly applied to the original problem, the result of computation is also correct. Such guarantee of correctness is usually not clearly discussed in the theory of term rewriting systems.

The rest of this paper is organized as follows: Section 2 gives an introductory example used throughout the paper. Section 3 reviews KR-Logic, a foundation to formalize the problem (Akama et al., 2019a). Section 4 introduces a class of conditional term rewriting rules and gives a theoretical bases for handling equality atoms in KR-Logic. Section 5 proves the correctness of the rule generation method for a given clause set. Section 6 shows how we can build a conditional term rewriting system for an introductory example by producing a set of term rewriting rules from a given clause set. Section 7 explains a solution built on the new conditional term rewriting system and the computation steps are shown for an example input. Section 8 concludes the paper.



Yoshida, T., Nantajeewarawat, E., Munetomo, M. and Akama, K.
Logical Approach to Theorem Proving with Term Rewriting on KR-logic.
DOI: 10.5220/0008165902820289
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 282-289
ISBN: 978-989-758-382-7

# 2 AN INTRODUCTORY EXAMPLE

We take a proof problem in the group theory and show intuitive solution by term rewriting. We will try to formalize this solution in later sections.

## 2.1 A Proof Problem of a Group

Consider the following definition of a group:

$\langle G, (\cdot), e, (^{-1}) \rangle$ is a group iff

1. $G$ is a set,
2. $(\cdot)$ is a mapping from $G \times G$ to $G$,
3. $e \in G$, and
4. $(^{-1})$ is a mapping from $G$ to $G$,

that satisfy the following conditions:

A1 (associativity) : $\forall x,y,z \in G : x \cdot (y \cdot z) = (x \cdot y) \cdot z$
A2 (identity) : $\forall x \in G : x \cdot e = e \cdot x = x$
A3 (inverse) : $\forall x \in G, \exists x^{-1} \in G :$
$x \cdot x^{-1} = x^{-1} \cdot x = e$

Using these three axioms, we want to prove that $((a^{-1} \cdot a) \cdot (b \cdot b^{-1}))^{-1} = b \cdot ((a \cdot b)^{-1} \cdot a)$.

## 2.2 Informal Proof with Equalities

We formalize the proof problem in 2.1 by introducing two function-like notations. Assume that

1. $(\cdot)$ is a mapping $f$ from $G \times G$ to $G$, and
2. $(^{-1})$ is a mapping $i$ from $G$ to $G$.

Using these mappings, we have the following five equations corresponding to three axioms in 2.1.

e1: $f(x, f(y,z)) = f(f(x,y),z)$
e2: $f(x,e) = x$
e3: $f(e,x) = x$
e4: $f(x,i(x)) = e$
e5: $f(i(x),x) = e$

Let $a, b$ be a term in $G$. Let's take an couple of terms $i(f(f(i(a),a), f(b,i(b))))$ as $((a^{-1} \cdot a) \cdot (b \cdot b^{-1}))^{-1}$ and $f(b, f(i(f(a,b)),a)$ as $b \cdot ((a \cdot b)^{-1} \cdot a)$. The original proof problem is then considered to determine the equality of these two terms.

In addition to the set of equalities above, we have four additional equalities derived from the original equalities.

e6: $i(e) = e$
e7: $f(x, f(y, i(f(x,y)))) = e$
e8: $f(y, i(f(x,y))) = i(x)$
e9: $i(f(y,x)) = f(i(x), i(y))$

For example, a generation process of e7 is shown as follows:

$f(x,i(x)) = e$ (p1)
$\Rightarrow f(f(x,y), i(f(x,y))) = e$ (p2)
$\Rightarrow f(x, f(y, i(f(x,y)))) = e$ (e7)

Starting with p1, a substitution $\{x/f(x,y)\}$ is applied to p1 then p2 is obtained. Applying e1 to p2 results in e7. In similar way, e6 is obtained. Using e7, e8 and e9 are generated.

By replacement of the terms using these nine equalities, each term reaches $e$. This obviously means that these two are equal;

$i(f(f(i(a),a), f(b,i(b)))) = f(b, f(i(f(a,b)),a)$.

## 2.3 How to Give Correctness to the Informal Method

The informal method is only procedural, i.e.,

1. each term rewriting rule is constructed without proving it to be semantically correct.
2. each successive transformation process is obtained by application of rewriting rules without correctness of computation result.

Our theory is correctness-based, i.e.,

1. each term rewriting rule is proved to be an ET rule.
2. each successive transformation process is obtained with guarantee of correctness of computation result.

KR-logic is sufficient for correctness-based theory, while first order logic is not.

# 3 REPRESENTATION IN KR-LOGIC

We use *KR-Logic* in order to formalize the original problem in 2.1 as a proof problem on a logical structure. *KR-Logic* is considered as an extension of usual first order logic by introduction of existential quantification of function variables, which is essential for representation of proof problems with equality constraints.

## 3.1 KR-logic

We take $\text{ECLS}_\text{N}$ as a logical structure $\mathcal{L}$, where $\text{ECLS}_\text{N}$ is the space of all clauses (including non-flat ones) on KR-logic. It is an extension of $\text{ECLS}_\text{F}$ (Akama and Nantajeewarawat, 2018) that contain only flat clauses, which is also an extension of CLS consisting of usual clauses.

Let $K$ denote the conjunction of the following formulas:

$F_1 : \forall x \forall y : (g(x) \wedge g(y) \rightarrow g(\$f(x,y)))$
$F_2 : \forall x : (g(x) \rightarrow g(\$i(x)))$
$F_3 : g(\$e)$
$F_4 : \forall x \forall y \forall z : ((g(x) \wedge g(y) \wedge g(z)) \rightarrow eq(\$f(x,\$f(y,z)),\$f(\$f(x,y),z)))$
$F_5 : \forall x : (g(x) \rightarrow eq(\$f(x,\$e),x))$
$F_6 : \forall x : (g(x) \rightarrow eq(\$f(\$e,x),x))$
$F_7 : \forall x : (g(x) \rightarrow eq(\$f(x,\$i(x)),\$e))$
$F_8 : \forall x : (g(x) \rightarrow eq(\$f(\$i(x),x),\$e))$

These formulas give definitions of $\$f$, $\$i$, and $\$e$ which are corresponding to conditions listed in 2.1.

## 3.2 Formalization in Clausal Form in KR-logic

Let term $t_a$ and $t_b$ be

$t_a = \$i(\$f(\$f(\$i(x),x),\$f(y,\$i(y))))$ and
$t_b = \$f(y,\$f(\$i(\$f(x,y)),x)$,

respectively. In order to ensure the equality of these two terms, we want to prove

$$K \rightarrow (\forall x \forall y : (g(x) \wedge g(y)) \rightarrow eq(t_a,t_b)).$$

Its negation is

$$K \wedge \neg(\forall x \forall y : (g(x) \wedge g(y)) \rightarrow eq(t_a,t_b)),$$

which is equivalent to

$$K \wedge (\exists x \exists y : (g(x) \wedge g(y) \wedge \neg eq(t_a,t_b))).$$

By meaning-preserving Skolemization (MPS), this formula is transformed into

$$E : \exists \$f \exists \$i \exists \$e \exists \$a \exists \$b : K \wedge g(\$a) \wedge g(\$b) \wedge \neg eq(t_a,t_b),$$

where $\$a$ and $\$b$ are new function variables.

Each formula in $K$ is transformed into a set of clausal forms using MPS as follows: $MPS(F_1) = C_1$, $MPS(F_2) = C_2$, $MPS(F_3) = C_3$, $MPS(F_4) = C_4$, $MPS(F_5) = C_5$, $MPS(F_6) = C_6$, $MPS(F_7) = C_7$, and $MPS(F_8) = C_8$.

The formula $E$ is eventually represented in a clausal form by:

$C_0$: $\leftarrow eq(\$i(\$f(\$f(\$i(\$a),\$a),\$f(\$b,\$i(\$b)))), \$f(\$b,\$f(\$i(\$f(\$a,\$b)),\$a)))$
$C_1$: $g(\$f(x,y)) \leftarrow g(x),g(y)$
$C_2$: $g(\$i(x)) \leftarrow g(x)$
$C_3$: $g(\$e) \leftarrow$
$C_4$: $eq(\$f(x,\$f(y,z)),\$f(\$f(x,y),z)) \leftarrow g(x),g(y),g(z)$
$C_5$: $eq(\$f(x,\$e),x) \leftarrow g(x)$
$C_6$: $eq(\$f(\$e,x),x) \leftarrow g(x)$
$C_7$: $eq(\$f(x,\$i(x)),\$e) \leftarrow g(x)$
$C_8$: $eq(\$f(\$i(x),x),\$e) \leftarrow g(x)$
$C_9$: $g(\$a) \leftarrow$
$C_{10}$: $g(\$b) \leftarrow$

Let $Cs$ be a set of clauses in $\text{ECLS}_\text{N}$ and equal to $\{C_1,C_2,\ldots,C_{10}\}$. Then we solve the given proof problem by proving that $Models(\{C_0\} \cup Cs) = \emptyset$.

# 4 TERM REWRITING RULES

We introduce a class of *conditional term rewriting rules* based on the equality atoms in a given clause set.

## 4.1 Overview

We introduce a logical structure $\text{ECLS}_\text{N}$, in which an equational clause is defined. A conditional term rewriting rule is defined. We obtain a conditional term rewriting rule from an equational clause. Rewriting relation determined by a conditional term rewriting rule is defined. We propose a sufficient condition of the correctness of a conditional term rewriting rule that is obtained from an equational clause.

## 4.2 Alphabet and Terms of KR-logic

An alphabet $\langle \mathbb{F}, \mathbb{V}, \mathbb{FC}, \mathbb{FV}, Pred, Pred_\text{C} \rangle$ is assumed, where $\mathbb{F}$ is a set of constructors, $\mathbb{V}$ is a set of usual variables, $\mathbb{FC}$ is a set of function constants, $\mathbb{FV}$ is a set of function variables, $Pred$ is a set of user-defined predicate symbols, and $Pred_\text{C}$ is a set of built-in constraint predicate symbols. Each element in $\mathbb{F} \cup \mathbb{FC} \cup \mathbb{FV}$ is associated with a non-negative integer, called its arity.

**Definition 1.** A *term* on $\langle \mathbb{F}, \mathbb{V}, \mathbb{FC}, \mathbb{FV} \rangle$, which is also simply called a term, is inductively defined as follows:

1. A 0-ary element in $\mathbb{F} \cup \mathbb{FC}$ is a term.
2. If $v \in \mathbb{V} \cup \mathbb{FV}$, then $v$ is a term.
3. If $f \in \mathbb{F} \cup \mathbb{FC} \cup \mathbb{FV}$, the arity of $f$ is $n > 0$, and $t_1,\ldots,t_n$ are terms, then $f(t_1,\ldots,t_n)$ is a term. □

The set of all terms on $\langle \mathbb{F}, \mathbb{V}, \mathbb{FC}, \mathbb{FV} \rangle$ is denoted by $\text{T}(\mathbb{F}, \mathbb{V}, \mathbb{FC}, \mathbb{FV})$. Let $\text{T}(\mathbb{F}) = \text{T}(\mathbb{F}, \emptyset, \emptyset, \emptyset)$. Let $\text{T}(\mathbb{F}, \mathbb{FC}) = \text{T}(\mathbb{F}, \emptyset, \mathbb{FC}, \emptyset)$. A term in $\text{T}(\mathbb{F}, \mathbb{FC})$ is called a *ground term*.

### 4.3 Term Contexts, Atom Contexts, and Clause Contexts

Assume that $\Box$ is a function symbol with arity 0 that does not belong to $\mathbb{F} \cup \mathbb{V} \cup \mathbb{FC} \cup \mathbb{FV}$. A subterm is a part of a term, and a subterm is extended by a *term-context* into a term. Let $t$ be a term and $tc$ a term context. $t \rhd tc$ is the term $tc\{\Box/t\}$. A term is a part of an atom, and a term is extended by an *atom context* into an atom. Let $t$ be a term and $ac$ an atom context. $t \rhd ac$ is the atom $ac\{\Box/t\}$. An atom is a part of a clause, and an atom is extended by a *clause context* into a clause. Let $a$ be an atom and $con$ a clause context. $a \rhd con$ is the clause $con\{\Box/a\}$. The sets of all *term contexts*, all *atom contexts*, and all *clause contexts* are denoted, respectively, by $Con_T$, $Con_A$, and $Con_C$.

### 4.4 Conditional Term Rewriting Rules

#### 4.4.1 Equational Clauses

An equational clause is a clause in $\mathrm{ECLS_N}$ of the form

$$eq(t_1, t_2) \leftarrow conds,$$

where $t_1$ and $t_2$ are terms in $\mathrm{T}(\mathbb{F}, \mathbb{V}, \mathbb{FC}, \mathbb{FV})$, and *conds* is a finite sequence of atoms. The set of all equational clauses in $\mathrm{ECLS_N}$ is denoted by $EQC$.

#### 4.4.2 Conditional Term Rewriting Rule

A *conditional term rewriting rule*, "CTRR" for short, is a formula of the form $(t_1 \rightarrow t_2 : conds)$. A *conditional term rewriting rule* $r$ of the formula $(t_1 \rightarrow t_2 : conds)$ is often represented by $r : (t_1 \rightarrow t_2 : conds)$. The set of all *conditional term rewriting rules* is denoted by $CTR$.

#### 4.4.3 Equational Clause to CTRR

An equational clause $C = (eq(t_1, t_2) \leftarrow conds)$ determines a set of two *conditional term rewriting rules*, which is denoted by $ctrs(C)$, i.e.,

$$ctrs(C) = \{(t_1 \rightarrow t_2 : conds), (t_2 \rightarrow t_1 : conds)\}.$$

### 4.5 CTRR as Relation of Clauses

A *conditional term rewriting rule* $r$ determines a relation on $pow(\mathrm{ECLS_N})$, denoted by $rel(r)$, as follows:

**Definition 2.** Let $r$ be a conditional term rewriting rule. Let $\mathcal{S}_\theta$ be the set of all substitutions on $\mathbb{V}$.

$$\begin{aligned} rel(r) = \{\ (C_1, C_2) \mid\ & r = (t_1 \rightarrow t_2 : conds) \\ & \&\ (tc \in Con_T)\ \&\ (ac \in Con_A) \\ & \&\ (con \in Con_C) \\ & \&\ (\theta \in \mathcal{S}_\theta)\ \&\ (conds\theta = true) \\ & \&\ (C_1 = ((((t_1\theta) \rhd tc) \rhd ac) \rhd con)) \\ & \&\ (C_2 = ((((t_2\theta) \rhd tc) \rhd ac) \rhd con))\ \}. \end{aligned}$$

Figure 1 illustrates how the element of $rel(r)$ is constructed by a couple of clauses $C_1$ and $C_2$. Assume that $\theta$ is given to fullfil the conditions denoted by $conds\theta$, all occurrences of $t_1\theta$ are valid under the term context $tc$, with which all atom occurrences containing $t_1\theta$ are valid under the atom context $ac$, in which a clause containing the atoms with $t_1\theta$ is valid under the clause context $con$. The same validity is confirmed for $t_2\theta$ in $C_2$. Then an element of $rel(r)$ is obtained as a pair of $C_1$ and $C_2$.

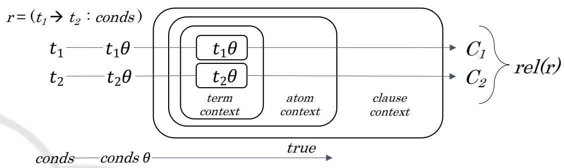


Figure 1: Element of $rel(r)$ visualized.

**Definition 3.** $C_1$ is transformed into $C_2$ by a conditional term rewriting rule $r$, denoted by $C_1 \xrightarrow{r} C_2$, iff $(C_1, C_2) \in rel(r)$.

### 4.6 Basic Theorem

**Theorem 1.** Let $Cs$ be a set of clauses. Let $C$, $C_1$, and $C_2$ be clauses in $\mathrm{ECLS_N}$. If $Models(Cs) = Models(Cs \cup \{C\})$, $ctrs(C) \ni r$, then $r$ is model-preserving. $\Box$

*Proof.* According to Theorem 1 in (Akama et al., 2019b), obviously $Models(Cs \cup \{C_1\}) = Models(Cs \cup \{C_2\})$. $\Box$

## 5 MAKING CORRECT TERM REWRITING RULE

We introduce a mechanism to generate a new conditional term rewriting rule from a set of clauses. Also, we prove a theorem for giving the correctness of generated rules.

### 5.1 Generation of CTRR

We describe the method of generating a CTRR after transforming a set of equational clauses. A transformation is caused by two ways, i.e., substitution and rewriting.

**Definition 4.** Let $Cs$ be a set of clauses. Let $C$ be a clause in $\mathrm{ECLS_N}$. Let $R$ be a set of term rewriting rules. $Cs \overset{R}{\rightsquigarrow} C$ is defined inductively by:

- If $Cs \ni C$, then $Cs \overset{R}{\rightsquigarrow} C$.
- If $Cs \overset{R}{\rightsquigarrow} C'$, $\theta$ is a substitution, and $C'\theta = C$, then $Cs \overset{R}{\rightsquigarrow} C$.
- If $Cs \overset{R}{\rightsquigarrow} C'$, $r \in R$, and $C' \xrightarrow{r} C$, then $Cs \overset{R}{\rightsquigarrow} C$.

Using above relation, we can make a rewriting rule.

**Definition 5.** $Cs \overset{R}{\mapsto} r$ iff there is a clause $C$ such that $Cs \overset{R}{\rightsquigarrow} C$ and $ctrs(C) \ni r$.

### 5.2 Main Theorem for Generation

A logical consequence relation, $lcr$ is defined by $lcr(X) = (Models(Cs) = Models(Cs \cup \{X\}))$, where $X$ is an equational clause. A transformation by substitution preserves a logical consequence relation.

**Proposition 1.** Let $Cs$ be a subset of $\mathrm{ECLS_N}$, and $C$ a clause in $\mathrm{ECLS_N}$. Let $\theta$ be a substitution. If $Models(Cs) = Models(Cs \cup \{C\})$, then $Models(Cs) = Models(Cs \cup \{C\theta\})$.

*Proof.* Since (1) $G \in Models(Cs)$ iff $G \in Models(Cs \cup \{C\})$, and (2) $G \in Models(Cs \cup \{C\})$ iff $G \in Models(Cs \cup \{C\theta\})$, we have $G \in Models(Cs)$ iff $G \in Models(Cs \cup \{C\theta\})$. Hence $Models(Cs) = Models(Cs \cup \{C\theta\})$. □

A logical consequence relation is preserved by repeated application of specialization and rewriting.

**Proposition 2.** Let $Cs$ be a set of clauses. Let $C$ be a clause in $\mathrm{ECLS_N}$. Assume that $R$ is a set of model-preserving term rewriting rules. If $Cs \overset{R}{\rightsquigarrow} C$, then $Models(Cs) = Models(Cs \cup \{C\})$. □

*Proof.* Assume that $Cs \overset{R}{\rightsquigarrow} C$.

- (base case) Assume that $Cs \ni C$. Then, $Cs = Cs \cup \{C\}$. Hence, $Models(Cs) = Models(Cs \cup \{C\})$.
- (inductive case) There is a clause $C'$, a substitution $\theta$, and $r \in R$ such that (1) $Cs \overset{R}{\rightsquigarrow} C'$, and (2) $C'\theta \xrightarrow{r} C$. By the inductive hypothesis and (1), $Models(Cs) = Models(Cs \cup \{C'\})$. By proposition 1, $Models(Cs) = Models(Cs \cup \{C'\theta\})$. By $r \in R$ and (2), $Models(Cs \cup \{C'\theta\}) = Models(Cs \cup \{C\})$. Hence, $Models(Cs) = Models(Cs \cup \{C\})$.

□

The method of rule generation gives model-preserving rules.

**Theorem 2.** Let $Cs$ be a subset of $\mathrm{ECLS_N}$. Let $R$ be a set of term rewriting rules. Let $r$ be a conditional term rewriting rule such that $Cs \overset{R}{\mapsto} r$. If $R$ is model-preserving, then $r$ is also model-preserving.

*Proof.* $Models(Cs) = Models(Cs \cup \{C_1\})$ and from Theorem 1, $Models(Cs \cup \{C_1\}) = Models(Cs \cup \{C_2\})$. Then $Models(Cs) = Models(Cs \cup \{C_2\})$. □

A generated rule is model-preserving since all transformations in the generation sequence and a rule generation method preserve a logical consequence relation.

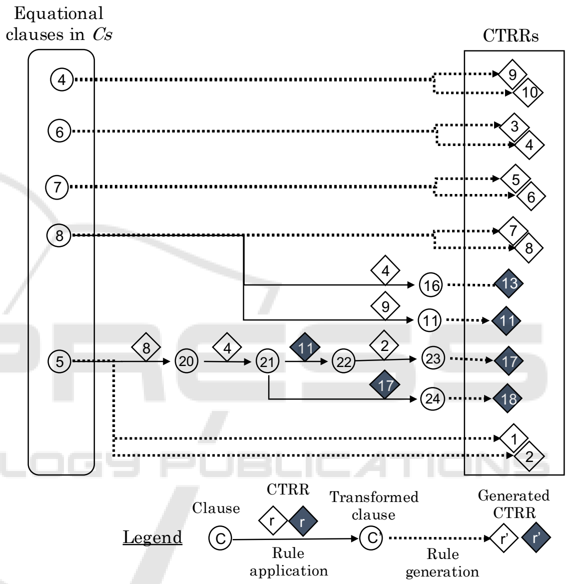


Figure 2: CTRRs expansion by rule generation.

## 6 CONSTRUCTING TERM REWRITING SYSTEM WITH RULE GENERATION

We introduce a method to construct a term rewriting system which equips rule generation.

### 6.1 Term Rewriting System Overview

Once a logical structure containing equational clauses are constructed, then we can develop a term rewriting system which contains a set of CTRRs growing by rule generation.

Figure 2 shows the overview of CTRRs expansion. The basic diagram elements shown in Legend are:

- Clause, a circled number represents an equational clause with its number
- Rule application, a solid line between two circles represents a transformation using pre-generated CTRR
- CTRR, a diamond with inner number. A white diamond represents a simple CTRR, while a dark diamond a CTRR from a transformed clause, and
- Rule generation, a dotted line between a clause and a CTRR

CTRR #1 to #10 are generated in straightforward way as shown in 4.4.3. For example, the top diamond in "CTRRs" section at the right side means $r_9$, which is generated from $C_4$. It is denoted as $ctrs(C_4) \ni r_9$. CTRR #11, #13, #17, and #18 are generated from transformed clauses by applying one or more pre-generated CTRRs. For example, the negative diamond of 13 is a generated rule from $C_{16}$ which is a resulting clause of rule application to $C_8$.

As a result, a set of CTRRs are growing by generating a new one from a transformed clause using existing CTRRs.

## 6.2 Generation of Term Rewriting Rules

Figure 2 shows the mechanism where conditional term rewriting rules are generated from a set of equational clauses. In "Equational clauses in *Cs*" section at the left side of Figure 2, $C_4$, $C_6$, $C_7$, $C_8$, and $C_5$ are listed in this order. For example, the circle of #8 in "Equational clauses in *Cs*" is connected to the circle of #16 with a solid line and the diamond of #4, which means that by application of $r_4$, $C_8$ is transformed into $C_{16}$. Then the dark diamond of #13 is generated from a transformed clause #16.

## 6.3 Rule Generation Sequences

In Figure 2, we have 14 generation lines. Each line consists of a rule generation arrow line and zero or more rule application arrow lines. When we represent a rule generation line using $\rightsquigarrow$, $ctrs$, and $\mapsto$, we call such formal representation a "rule generation sequence." Here are 14 generation sequences appear in Figure 2.

#4: $ctrs(C_4) = \{r_9, r_{10}\}$
#6: $ctrs(C_6) = \{r_3, r_4\}$
#7: $ctrs(C_7) = \{r_5, r_6\}$
#8(a): $ctrs(C_8) = \{r_7, r_8\}$
#8(b): $Cs \rightsquigarrow C_{16}$, $ctrs(C_{16}) \ni r_{13}$;
#8(c): $Cs \rightsquigarrow C_{11}$, $ctrs(C_{11}) \ni r_{11}$;
#5(a): $Cs \rightsquigarrow C_{23}$, $ctrs(C_{23}) \ni r_{17}$;
#5(b): $Cs \rightsquigarrow C_{24}$, $ctrs(C_{24}) \ni r_{18}$;
#5(c): $ctrs(C_5) = \{r_1, r_2\}$

## 6.4 Generation of $r_{17}$ from *Cs*

The following steps explain the generation sequence #5(a) for $r_{17}$ from *Cs* starting with $C_5$.

(1) $Cs \ni C_5$
(2) $C_5\theta_{20} \xrightarrow{r_8} C_{20}$
(3) $C_{20}\theta_{21} \xrightarrow{r_4} C_{21}$
(4) $C_{21}\theta_{22} \xrightarrow{r_{11}} C_{22}$
(5) $C_{22}\theta_{23} \xrightarrow{r_2} C_{23}$
(6) $ctrs(C_{23}) \ni r_{17}$

The 4th step is detailed as follows:

(4-1) Take $C_{21}$
$C_{21}$: $eq(y, \$f(\$i(x), \$f(x,y))) \leftarrow g(x), g(y)$
(4-2) Specialization by $\theta_{22} = \{y/\$f(y, \$i(\$f(x,y)))\}$
$$C_{21}\theta_{22}: eq(\$f(y, \$i(\$f(x,y))), \$f(\$i(x), \$f(x, \$f(y, \$i(\$f(x,y)))))) \leftarrow g(x), g(y)$$
(4-3) Application of $r_{11}$; $C_{21}\theta_{22} \xrightarrow{r_{11}} C_{22}$
$$r_{11}: (\$f(x, \$f(y, \$i(\$f(x,y)))) \to \$e : \{g(x), g(y)\})$$
$$C_{22}: eq(\$f(y, \$i(\$f(x,y))), \$f(\$i(x), \$e)) \leftarrow g(x), g(y)$$

## 6.5 Generated Rules

From all 14 generation sequences, we have conditional term rewriting rules as follows:

$r_1: (x \to \$f(x, \$e) : \{g(x)\})$
$r_2: (\$f(x, \$e) \to x : \{g(x)\})$
$r_3: (x \to \$f(\$e, x) : \{g(x)\})$
$r_4: (\$f(\$e, x) \to x : \{g(x)\})$
$r_5: (\$e \to \$f(x, \$i(x)) : \{g(x)\})$
$r_6: (\$f(x, \$i(x)) \to \$e : \{g(x)\})$
$r_7: (\$e \to \$f(\$i(x), x) : \{g(x)\})$
$r_8: (\$f(\$i(x), x) \to \$e : \{g(x)\})$
$r_9: (\$f(\$f(x,y), z) \to \$f(x, \$f(y,z)) : \{g(x), g(y), g(z)\})$
$r_{10}: (\$f(x, \$f(y,z)) \to \$f(\$f(x,y), z) : \{g(x), g(y), g(z)\})$
$r_{11}: (\$f(x, \$f(y, \$i(\$f(x,y)))) \to \$e) : \{g(x), g(y)\})$
$r_{13}: (\$i(\$e) \to \$e : \{\})$
$r_{17}: (\$f(y, \$i(\$f(x,y))) \to \$i(x) : \{g(x), g(y)\})$
$r_{18}: (\$i(\$f(y,x)) \to \$f(\$i(x), \$i(y)) : \{g(x), g(y)\})$

# 7 LPSF-BASED SOLUTION WITH TERM REWRITING

We propose a LPSF-based solution for problems using term rewriting. First, we give a theoretical foundation of the correctness of the solution, followed by the solution for the sample problem.

## 7.1 Clause-rule Interaction Tree

A clause is transformed into a clause by repeated application of rules, which is represented by a triple.

$C \circ [r_1, ..., r_n] = C'$ is defined inductively by:

- $C \circ [] = C$.
- $C \circ [r_1, ..., r_{n-1}] = C''$, and there is $\theta$ such that $C''\theta \overset{r_n}{\to} C'$.

We add a generated rule to a triple to form a quadruple.

**Definition 6.** Let $Cs$ a set of clauses. Clause-Rule Interaction quadruple wrt $Cs$ is a tuple of the form $(C, [r_1, ..., r_n], C', r')$, where $C$ and $C'$ are clauses, $r_1, ..., r_n$ and $r'$ are term rewriting rules, $C \in Cs$, $C \circ [r_1, ..., r_n] = C'$, and $ctrs(C') \ni r'$.

Notations for an element designator in a sequence and a subsequence are introduced.

**Definition 7.** Let $X$ be a finite sequence. $nth(m,X)$ is defined as the $m$-th element of $X$. $Under(m,X)$ is defined as the set of all elements of $X$ until $m$-th element, i.e., $Under(m,X) = \{nth(1,X), nth(2,X), \ldots, nth(m,X)\}$.

Two notations for obtaining clauses and rules from a set of CRI quadruples.

**Definition 8.** Let $X$ be a set of Clause-Rule Interaction quadruples. We define $cl(X)$ and $trs(X)$ by:

- $cl(X) = \{C \mid (C, [r_1, ..., r_n], C', r') \in X\}$.
- $trs(X) = \{r' \mid (C, [r_1, ..., r_n], C', r') \in X\}$.

If a rule is applied for rule generation, the rule must be already generated.

**Definition 9.** Let $Cs$ a set of clauses. Clause-Rule Interaction Tree wrt $Cs$, denoted by $CRIT(Cs)$, is a finite sequence of clause-rule interaction quadruples wrt $Cs$ such that if $(C, [r_1, ..., r_n], C', r') = nth(m, CRIT(Cs))$, then $\{r_1, ..., r_n\} \subseteq trs(Under(m-1, CRIT(Cs)))$.

Example:

$$
\begin{aligned}
CRIT(Cs) = (\ & (C_4, [], C_4, r_9), \\
& (C_4, [], C_4, r_{10}), \\
& (C_5, [], C_5, r_1), \\
& (C_5, [], C_5, r_2), \\
& \ldots \\
& (C_8, [r_4], C_{16}, r_{13}), \\
& (C_8, [r_9], C_{11}, r_{11}), \\
& (C_5, [r_8, r_4, r_{11}, r_2], C_{23}, r_{17}), \\
& (C_5, [r_8, r_4, r_{17}], C_{24}, r_{18})\ )
\end{aligned}
$$

## 7.2 Correctness

Rules generated in a Clause-Rule Interaction Tree are model-preserving.

**Theorem 3.** Let $Cs$ be a set of clauses. Let $r$ be a conditional term rewriting rule. If $trs(CRIT(Cs)) \ni r$, then $r$ is model-preserving. □

*Proof.* We prove the theorem by induction of the size of $CRIT(Cs)$. Assume that $trs(CRIT(Cs)) \ni r$.

- When $size(CRIT(Cs)) = 1$. Then $(C, [], C', r) = nth(1, CRIT(Cs))$. Hence $C = C'$ and $Models(Cs) = Models(Cs \cup \{C'\})$. Let $R' = \{\}$. Since $Cs \overset{R'}{\rightsquigarrow} C'$ and $ctrs(C') \ni r$, we have $Cs \overset{R'}{\mapsto} r$. By Theorem 2, $r$ is model-preserving.
- When $size(CRIT(Cs)) > 1$. Assume that the theorem holds for $Under(m-1, CRIT(Cs))$. Let $R = trs(Under(m-1, CRIT(Cs)))$. Each $r' \in R$ is model-preserving. Assume also that $(C, [r_1, ..., r_n], C', r) = nth(m, CRIT(Cs))$. Let $R' = \{r_1, ..., r_n\}$. Then $Cs \overset{R'}{\rightsquigarrow} C'$. Since $CRIT(Cs)$ is a clause-rule interaction tree wrt $Cs$, we have $R' \subseteq R$. Hence $R'$ is model-preserving. Since $Cs \overset{R'}{\rightsquigarrow} C'$ and Proposition 2, we have $Models(Cs) = Models(Cs \cup \{C'\})$. Since $Cs \overset{R'}{\rightsquigarrow} C'$ and $ctrs(C') \ni r$, we have $Cs \overset{R'}{\mapsto} r$. By Theorem 2, $r$ is model-preserving.

□

## 7.3 Solution for the Sample Problem

The sample proof problem is proven by the application of model-preserving term rewriting rules generated in this paper. All steps of computation are shown below. $\mathcal{M}$ is used as a short-hand for *Models* for saving spaces.

$$
\begin{aligned}
& \mathcal{M}(\{(\leftarrow eq(\$i(\$f(\underline{\$f(\$i(\$a), \$a)}, \$f(\$b, \$i(\$b)))), \\
& \qquad \$f(\$b, \$f(\$i(\$f(\$a, \$b)), \$a))))\} \cup Cs) \\
& \quad (\text{Apply } r_8 \text{ with } \theta_{31} = \{x/\$a\}) \\
= \ & \mathcal{M}(\{(\leftarrow eq(\$i(\$f(\$e, \$f(\$b, \$i(\$b)))), \\
& \qquad \$f(\$b, \$f(\underline{\$i(\$f(\$a, \$b))}, \$a))))\} \cup Cs) \\
& \quad (\text{Apply } r_{18} \text{ with } \theta_{32} = \{x/\$a, y/\$b\} \\
= \ & \mathcal{M}(\{(\leftarrow eq(\$i(\$f(\$e, \underline{\$f(\$b, \$i(\$b))})), \\
& \qquad \$f(\$b, \$f(\$f(\$i(\$b), \$i(\$a)), \$a))))\} \cup Cs) \\
& \quad (\text{Apply } r_6 \text{ with } \theta_{33} = \{x/\$b\})
\end{aligned}
$$

$$
\begin{aligned}
&= \mathcal{M}(\{(\leftarrow eq(\$i(\$f(\$e,\$e)),\\
&\qquad \$f(\$b,\underline{\$f(\$f(\$i(\$b),\$i(\$a)),\$a)})))\} \cup Cs)\\
&\quad (\text{Apply } r_9 \text{ with } \theta_{34} = \{x/\$i(\$b), y/\$i(\$a), z/\$a\}\\
&= \mathcal{M}(\{(\leftarrow eq(\$i(\underline{\$f(\$e,\$e)}),\\
&\qquad \$f(\$b,\$f(\$i(\$b),\$f(\$i(\$a),\$a)))))\} \cup Cs)\\
&\quad (\text{Apply } r_4 \text{ with } \theta_{35} = \{x/\$e\})\\
&= \mathcal{M}(\{(\leftarrow eq(\$i(\$e),\\
&\qquad \$f(\$b,\$f(\$i(\$b),\underline{\$f(\$i(\$a),\$a)})))))\} \cup Cs)\\
&\quad (\text{Apply } r_8 \text{ with } \theta_{36} = \{x/\$a\})\\
&= \mathcal{M}(\{(\leftarrow eq(\underline{\$i(\$e)},\\
&\qquad \$f(\$b,\$f(\$i(\$b),\$e))))\} \cup Cs)\\
&\quad (\text{Apply } r_{13} \text{ with } \theta_{37} = \{\})\\
&= \mathcal{M}(\{(\leftarrow eq(\$e,\$f(\$b,\underline{\$f(\$i(\$b),\$e)})))\} \cup Cs)\\
&\quad (\text{Apply } r_2 \text{ with } \theta_{38} = \{x/\$i(\$b)\}\\
&= \mathcal{M}(\{(\leftarrow eq(\$e,\underline{\$f(\$b,\$i(\$b))}))\} \cup Cs)\\
&\quad (\text{Apply } r_6 \text{ with } \theta_{39} = \{x/\$b\})\\
&= \mathcal{M}(\{(\leftarrow eq(\$e,\$e))\} \cup Cs)\\
&\quad (\text{Remove the } eq(\$e,\$e) \text{ atom since it is true.})\\
&= \mathcal{M}(\{(\leftarrow)\} \cup Cs)\\
&= \emptyset.
\end{aligned}
$$

In the last step of transformation, a constraint solving rule for equality is used. This rule is not a term rewriting rule. However, it obviously preserves models. In our computation model, rules of logical inference and term rewriting can be used together under the principle of equivalent transformation.

# 8 CONCLUDING REMARKS

Term rewriting rules are used for computation when the background knowledge contains equational clauses. We can generate two term rewriting rules directly from an equational clause. We apply to an equational clause repeatedly (1) specialization by a substitution for usual variables, and (2) application of an already derived rewriting rule. We make term rewriting rules directly from the resulting equational clauses. We have proved that the obtained term rewriting rule is an ET rule. Since ET rules are repeatedly applied to the original problem, the result of computation is also correct. Such guarantee of correctness is usually not clearly discussed in the theory of term rewriting systems.

The informal method is procedural, i.e., semantical structure is not discussed relating to rewriting rules. Each term rewriting rule is constructed without proving it to be semantically correct. Moreover, correctness of computation result is guaranteed based on the correctness of each rewriting rule.

First order logic is not sufficient for correctness-based theory since it doesn't have enough expressive power, while KR-logic is sufficient due to the existence of function variables.

The theory in this paper is constructed on LPSF, where KR-logic is used as a canonical logical structure, and term rewriting rules are used as ET rules. The resolution rule and the unfolding rule are typical instances of rules in the domain of logic, while term rewriting rules are typical instances of functional rewriting. Hence, logical inference and functional rewriting co-exist, both of them being instances of a broader concept of equivalent transformation.

# REFERENCES

Akama, K. and Nantajeewarawat, E. (2016). Model-intersection problems with existentially quantified function variables: Formalization and a solution schema. In *Proc. 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 2, pages 52–63, Porto, Portugal.

Akama, K. and Nantajeewarawat, E. (2018). Solving query-answering problems with constraints for function variables. *Springer International Publishing AG, part of Springer Nature 2018, N. T. Nguyen et al. (Eds.): ACIIDS 2018, LNAI 10751*, pages 36–47.

Akama, K., Nantajeewarawat, E., and Akama, T. (2019a). Logical problem solving framework. *Springer International Publishing AG, part of Springer Nature 2019, N. T. Nguyen et al. (Eds.): ACIIDS 2019, LNAI 11431*, pages 28–40.

Akama, K., Nantajeewarawat, E., and Akama, T. (2019b). Term rewriting that preserves models in kr-logic. *Springer International Publishing AG, part of Springer Nature 2019, N. T. Nguyen et al. (Eds.): ACIIDS 2019, LNAI 11431*, pages 41–52.

Bird, R. and Wadler, P. (1988). Prentice-Hall, Inc., New Jersey.

Dershowitz, N. and Jouannaud, J.-P. (1990). pages 243–320. MIT Press.