International Conference on Rewriting Techniques and Applications 2010 (Edinburgh), pp. 7-16
http://rewriting.loria.fr/rta/

AUTOMATED CONFLUENCE PROOF BY DECREASING DIAGRAMS BASED ON RULE-LABELLING

TAKAHITO AOTO [1]

[1] RIEC, Tohoku University, 2-1-1 Katahira, Sendai, Miyagi, 980-8577, Japan
E-mail address: aoto@nue.riec.tohoku.ac.jp
URL: http://www.nue.riec.tohoku.ac.jp/user/aoto/

Abstract. Decreasing diagrams technique (van Oostrom, 1994) is a technique that can be widely applied to prove confluence of rewrite systems. To directly apply the decreasing diagrams technique to prove confluence of rewrite systems, rule-labelling heuristic has been proposed by van Oostrom (2008). We show how constraints for ensuring confluence of term rewriting systems constructed based on the rule-labelling heuristic are encoded as linear arithmetic constraints suitable for solving the satisfiability of them by external SMT solvers. We point out an additional constraint omitted in (van Oostrom, 2008) that is needed to guarantee the soundness of confluence proofs based on the rule-labelling heuristic extended to deal with non-right-linear rules. We also present several extensions of the rule-labelling heuristic by which the applicability of the technique is enlarged.

1. Introduction

Confluent term rewriting systems form a basis of flexible computations and effective deductions for equational reasoning [2, 10]. Thus, confluence is considered to be one of the most important properties for *term rewriting systems* (*TRSs* for short). In contrast to the termination proof techniques, where automation of the techniques has been actively investigated, not much attention has been paid to *automation of confluence proving*. Motivated by such a situation, Aoto et.al. [1, 15] have started developing a fully-automated confluence prover ACP.

Decreasing diagrams technique [11] is a technique that can be widely applied to prove confluence of rewrite systems. Many confluence results are explained and are extended based on the decreasing diagrams criterion [11, 13, 14]. In [13], *rule-labelling heuristic* has been proposed to prove confluence of rewrite systems directly by the decreasing diagrams technique. In the rule-labelling heuristic, each rewrite step is labeled by the rewrite rule employed in that rewrite step—then the existence of the suitable ordering on labels ensures the confluence of the TRSs. In [1, 15], the implementation of decreasing diagrams techniques in ACP was left as a future work.

1998 ACM Subject Classification: D.3.1 [Programming Languages]: Formal Definitions and Theory; F.3.1 [Logics and Meanings of Programs]: Specifying and Verifying and Reasoning about Programs; F.4.2 [Mathematical Logic and Formal Languages]: Grammars and Other Rewriting Systems; I.2.2 [Artificial Intelligence]: Automatic Programming.
Key words and phrases: Confluence, Decreasing Diagrams, Automation, Term Rewriting Systems.

Proceedings of the 21st International Conference on Rewriting Techniques and Applications, Edinburgh, July, 2010
Editors: Christopher Lynch
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.RTA.2010.7

In this paper, we report a method to incorporate the confluence proof by the decreasing diagrams based on the rule-labelling heuristic into the automated confluence provers such as the ACP. More precisely, we show how conditions for ensuring confluence of term rewriting systems constructed based on the rule-labelling heuristic are encoded as *linear arithmetic constraints* suitable for solving the satisfiability of them by *external SMT solvers* (SAT modulo theories where the linear arithmetic is employed for the underlying theory). Furthermore, we point out an additional condition omitted in [13] that is needed to guarantee the soundness of confluence proofs based on the rule-labelling heuristic. We also present several extensions of the rule-labelling heuristic by which the applicability of the technique is enlarged. All methods are implemented and experiments are reported.

The remainder of the paper is organized as follows. In Sec. 2, we briefly explain notions and notations used in this paper. In Sec. 3, we explain the decreasing diagrams technique and present an encoding of confluence criteria based on the basic version of the rule-labelling heuristic. In Sec. 4, we explain an extension of rule-labelling heuristic for left-linear TRSs. Here we point out the necessity of an additional condition omitted in [13]. In Sec. 5, we present the encoding of the criterion explained in Sec. 4 with a natural generalization. In section 6, we present two further flexibilities that can be added to the heuristic. Sec. 7 reports an implementation and experiments. Sec. 8 concludes.

2. Preliminaries

This section briefly explains notions and notations used in this paper. For omitted definitions, we refer [2].

Abstract reduction system (*ARS* for short) $\mathcal{A} = \langle A, (\to_i)_{i \in I} \rangle$ consists of a set A and indexed relations $\to_i$ over A. For $J \subseteq I$, $\to_J = \bigcup_{i \in J} \to_i$ and $\to_I$ is abbreviated to $\to$ if no confusion arises. The reverse of $\to$ is denoted by $\leftarrow$. The reflexive transitive closure (reflexive closure, equivalence closure) of $\to$ is denoted by $\xrightarrow{*}$ (resp. $\xrightarrow{=}$, $\xleftrightarrow{*}$). We use $\circ$ for the composition of relations. We denote a quasi-order by $\succsim$, its strict part by $\succ$ and its equivalence part by $\simeq$. A quasi-order is *well-founded* if there exists no infinite descending chain $a_0 \succ a_1 \succ \cdots$. We put $\curlyvee m = \{i \mid i \prec m\}$, $\simeq m = \{i \mid i \simeq m\}$ and $\curlyvee l, m = \{i \mid i \prec l\} \cup \{i \mid i \prec m\}$. The lexicographic comparison $\succsim_{\text{lex}}$ by two quasi-orders $\succsim_1$ and $\succsim_2$ is given by $\langle a_1, a_2 \rangle \succsim_{\text{lex}} \langle b_1, b_2 \rangle$ iff either $a_1 \succ_1 b_1$ or $a_1 \simeq_1 b_1$ and $a_2 \succsim_2 b_2$.

The sets of *function symbols* and *variables* are denoted by $\mathcal{F}$ and V. Each function symbol f is equipped with a natural number arity(f), the *arity* of f. A *constant* is a function symbol with arity 0. We denote by $\mathrm{V}(t)$ the set of variables occurring in a term t. A variable $x \in \mathrm{V}(t)$ is said to have a *linear occurrence in t* (or x is *linear in t*) if there is only one occurrence of x in t. A term t is *linear* if all variables in $\mathrm{V}(t)$ are linear in t. A *position* in a term is denoted by a (possibly empty) sequence of positive integers. The empty sequence (i.e. the *root* position) is denoted by ϵ. If p is a position in a term t, the *subterm* of t at p is denoted by t/p and we write $t[s]_p$ the term obtained from t by replacing the subterm at p with a term s. A *context* is a term with a special constant $\Box$ (called a *hole*). A context C with precisely one occurrence of the hole is denoted by $C[\,]$. We write $C[\,]_p$ if $C[\,]/p = \Box$. An instance of t by a substitution σ is written as $t\sigma$.

Any *rewrite rule* $l \to r$ satisfies the conditions (1) $l \notin V$ and (2) $\mathrm{V}(r) \subseteq \mathrm{V}(l)$. Rewrite rules are identified modulo variable renaming. A rewrite rule $l \to r$ is *linear* (*left-linear*, *right-linear*) if l and r (l, r, respectively) are linear. A *term rewriting system* (*TRS* for short) is a finite set of rewrite rules. It is linear (left-linear, right-linear) if so are all rules.

There is a *rewrite step* $s \to t$ if there exist a context $C[\,]_p$, a substitution σ, and a rewrite rule $l \to r \in \mathcal{R}$ such that $s = C[l\sigma]_p$ and $t = C[r\sigma]_p$. The subterm occurrence of $l\sigma$ at p is the *redex occurrence* of this rewrite step; the occurrence of l (except variables) in s is called the *redex pattern* of this rewrite step. A rewrite sequence of the form $s \leftarrow u \to t$ is called a *peak*; the one of the form $s \xrightarrow{*} \circ \xleftarrow{*} t$ is a *joinable rewrite sequence*. Terms s and t are *joinable* if $s \xrightarrow{*} \circ \xleftarrow{*} t$. A TRS $\mathcal{R}$ is *confluent* or *Church–Rosser* if $s \xleftarrow{*} \circ \xrightarrow{*} t$ implies s and t are joinable.

Let s, t be terms whose variables are disjoint. The term s *overlaps* on t (at position p) when there exists a non-variable subterm $u = t/p$ of t such that u and s are unifiable. Let $l_1 \to r_1$ and $l_2 \to r_2$ be rewrite rules w.l.o.g. whose variables are disjoint. Suppose that l_1 overlaps on l_2 at position p. Let σ be the most general unifier of l_1 and l_2/p. Then the term $l_2[l_1]\sigma$ yields a *critical peak* $l_2[r_1]\sigma \leftarrow l_2[l_1]\sigma \to r_2\sigma$. The pair $\langle l_2[r_1]\sigma, r_2\sigma\rangle$ is called the *critical pair* obtained by the overlap of $l_1 \to r_1$ on $l_2 \to r_2$ at position p. In the case of self-overlap (i.e. when $l_1 \to r_1$ and $l_2 \to r_2$ are identical modulo renaming), we do not consider the case $p = \epsilon$. The set of critical pairs obtained by an overlap of $l_1 \to r_1$ on $l_2 \to r_2$ is denoted by $\mathrm{CP}(l_1 \to r_1, l_2 \to r_2)$. The set of critical pairs in a TRS $\mathcal{R}$ is denoted by $\mathrm{CP}(\mathcal{R})$.

3. Confluence by decreasing diagrams based on rule-labelling heuristic

An ARS $\mathcal{A} = \langle A, (\to_i)_{i \in I}\rangle$ is *locally decreasing* w.r.t. a well-founded quasi-order $\succsim$ if for any $s \leftarrow_l \circ \to_m t$ there exists $s \xrightarrow{*}_{\prec l} \circ \xrightarrow{=}_{\simeq m} \circ \xrightarrow{*}_{\prec l,m} \circ \xleftarrow{*}_{\prec l,m} \circ \xleftarrow{=}_{\simeq l} \circ \xleftarrow{*}_{\prec m} t$ [11]. In this paper, we use the following variant of decreasing diagrams criterion.

Proposition 3.1 (decreasing diagrams criterion [11]). *An ARS $\mathcal{A} = \langle A, (\to_i)_{i \in I}\rangle$ is confluent if $\mathcal{A}$ is locally decreasing w.r.t. a well-founded quasi-order order $\succsim$.* ∎

In [13], *rule-labelling heuristic* is introduced to apply the decreasing diagrams criterion to directly prove confluence of TRSs. To use the decreasing diagrams criterion, each rewrite step needs to be equipped with a label—in the rule-labelling heuristic, the rewrite rule employed in the rewrite step is used as the label of each rewrite step. As in [13], let us suppose that each rewrite rule is numbered from 1 to $|\mathcal{R}|$ and that each rewrite rule is identified with its number ($i : l \to r \in \mathcal{R}$ indicates that i is the number of $l \to r$). We say a peak $s \leftarrow_i u \to_j t$ is *locally decreasing w.r.t.* $\succsim$ if there is a rewrite sequence of the form $s \xrightarrow{*}_{\prec i} \circ \xrightarrow{=}_{\simeq j} \circ \xrightarrow{*}_{\prec i,j} \circ \xleftarrow{*}_{\prec i,j} \circ \xleftarrow{=}_{\simeq i} \circ \xleftarrow{*}_{\prec j} t$.

Proposition 3.2 (confluence by rule-labelling heuristic [13]). *A linear TRS $\mathcal{R}$ is confluent if there exists a quasi-order $\succsim$ on $\mathcal{R}$ such that any critical peak of $\mathcal{R}$ is locally decreasing w.r.t. $\succsim$.* ∎

Note that well-founded of $\succsim$ follows from the finiteness of the set $\mathcal{R}$. Based on this proposition, a (basic) confluence proof of TRS $\mathcal{R}$ by decreasing diagrams based on rule-labelling is conducted as follows.

Step 1 Check (left- and right-)linearity.

Step 2 Find a joinable rewrite sequence $s \xrightarrow{*} v \xleftarrow{*} t$ for every critical pairs $\langle s, t\rangle \in \mathrm{CP}(i, j)$.

Step 3 Check whether there exists a quasi-order $\succsim$ on $\mathcal{R}$ such that $s \xrightarrow{*} v \xleftarrow{*} t$ (obtained in the step 2) has the form $s \xrightarrow{*}_{\prec i} \circ \xrightarrow{=}_{\simeq j} \circ \xrightarrow{*}_{\prec i,j} \circ \xleftarrow{*}_{\prec i,j} \circ \xleftarrow{=}_{\simeq i} \circ \xleftarrow{*}_{\prec j} t$ for every critical pairs $\langle s, t\rangle \in \mathrm{CP}(i, j)$.

Computation of the step 1 is easy. Automation of the step 2 is partially achieved by imposing a maximum length on rewrite steps $s \xrightarrow{*} v \xleftarrow{*} t$. The only non-trivial part is the step 3. We show that the step 3 can be solved by reducing the problem into the satisfiability of an arithmetic constraint.

First, let us illustrate by an example how the requirement on the quasi-order $\precsim$ is specified. Since the requirements on $s \xrightarrow{*} v$ and $v \xleftarrow{*} t$ are symmetric, we concentrate on the $s \xrightarrow{*} v$ part. Suppose that we have a joinable rewrite sequence $s \rightarrow_{x_1} \circ \rightarrow_{x_2} \circ \rightarrow_{x_3} v \xleftarrow{*} t$ for $\langle s, t\rangle \in \mathrm{CP}(i,j)$. The requirement is that $\rightarrow_{x_1} \circ \rightarrow_{x_2} \circ \rightarrow_{x_3}$ has the form $\xrightarrow{*}_{\prec i} \circ \xrightarrow{=}_{\simeq j} \circ \xrightarrow{*}_{\prec i,j}$. We can think of five possibilities depending on where the rewrite step equivalent to j (virtually) appears:

(i) $x_1, x_2, x_3 \in \prec i, j$ (i.e. the rewrite step equivalent to j is placed before $\rightarrow_{x_1}$ step)
(ii) $x_1 \simeq j$ and $x_2, x_3 \in \prec i, j$ (i.e. the rewrite step equivalent to j is $\rightarrow_{x_1}$ step)
(iii) $x_1 \prec i$, $x_2 \simeq j$ and $x_3 \in \prec i, j$ (i.e. the rewrite step equivalent to j is $\rightarrow_{x_2}$ step)
(iv) $x_1, x_2 \prec i$, $x_3 \simeq j$ (i.e. the rewrite step equivalent to j is $\rightarrow_{x_3}$ step)
(v) $x_1, x_2, x_3 \prec i$ (i.e. the rewrite step equivalent to j is placed after $\rightarrow_{x_3}$ step)

The last possibility is redundant because of the first one; thus four possibilities remain. Since $x_k \in \prec i, j$ equals to $(x_k \prec i) \vee (x_k \prec j)$, we obtain the following requirement on the quasi-order $\precsim$.

$$\begin{array}{lll} & (x_1 \prec i \vee x_1 \prec j) \wedge (x_2 \prec i \vee x_2 \prec j) \wedge (x_3 \prec i \vee x_3 \prec j) & \text{from the case (i)} \\ \vee & (x_1 \simeq j) \wedge (x_2 \prec i \vee x_2 \prec j) \wedge (x_3 \prec i \vee x_3 \prec j) & \text{from the case (ii)} \\ \vee & (x_1 \prec i) \wedge (x_2 \simeq j) \wedge (x_3 \prec i \vee x_3 \prec j) & \text{from the case (iii)} \\ \vee & (x_1 \prec i) \wedge (x_2 \prec i) \wedge (x_3 \simeq j) & \text{from the case (iv)} \end{array}$$

In describing the requirement for the general case, the following assumption is assumed. Below, each rewrite sequence is supposed to be assigned by a sequence of labels as $s \xrightarrow{*}_{i_1 \cdots i_k} t$ for $s \rightarrow_{i_1} \circ \cdots \circ \rightarrow_{i_k} t$.

Assumption 3.3. *We assume that there exists a joinable rewrite sequence $s \xrightarrow{*}_{\sigma} \circ \xleftarrow{*}_{\rho} t$ for each critical pair $\langle s, t\rangle \in \mathrm{CP}(i,j)$. Given labelling of rewrite steps, the sequences σ and ρ of labels are denoted by $\mathrm{JL}(s,t)$ and $\mathrm{JR}(s,t)$, respectively.*

Definition 3.4. We define $\mathrm{Ldd}(i,j,x_1 \cdots x_n)$ and $\mathrm{LDD}(\mathcal{R})$ as follows.

$$\begin{aligned} \mathrm{Ldd}(i,j,x_1 \cdots x_n) &= (\textstyle\bigwedge_{1 \le l \le n}((x_l \prec i) \vee (x_l \prec j))) \vee \\ &\quad \textstyle\bigvee_{1 \le k \le n} \left[(\bigwedge_{1 \le l < k}(x_l \prec i)) \wedge (x_k \simeq j) \wedge (\bigwedge_{k < l \le n}((x_l \prec i) \vee (x_l \prec j)))\right] \\ \mathrm{LDD}(\mathcal{R}) &= \textstyle\bigwedge_{i,j \in \mathcal{R}} \bigwedge_{\langle s,t\rangle \in \mathrm{CP}(i,j)} \left(\mathrm{Ldd}(i,j,\mathrm{JL}(s,t)) \wedge \mathrm{Ldd}(j,i,\mathrm{JR}(s,t))\right) \end{aligned}$$

Theorem 3.5. *A linear TRS $\mathcal{R}$ is confluent if there exists a quasi-order $\succsim$ on $\mathcal{R}$ that satisfies* $\mathrm{LDD}(\mathcal{R})$. ∎

We next explain how the existence problem of a quasi-order $\succsim$ on $\mathcal{R}$ that satisfy $\mathrm{LDD}(\mathcal{R})$ is reduced to the satisfiability problem of an arithmetic constraint. The idea is to specify the quasi-order $\succsim$ by the assignment of natural number weights, that is, $i \succ j$ iff the rule i has the weight strictly larger than that of j. Here, note that since $\mathcal{R}$ is finite and the requirement is a monotone formula, it suffices to consider the total quasi-order. Suppose non-negative integer variables $w_1, \ldots, w_{|\mathcal{R}|}$ are to be assigned by the weight of the rules $1, \ldots, |\mathcal{R}| \in \mathcal{R}$. Then the requirement $\mathrm{LDD}(\mathcal{R})$ of $\succsim$ is translated to an arithmetic constraint $[\![\mathrm{LDD}]\!](\mathcal{R})$ over the indeterminates $w_1, \ldots, w_{|\mathcal{R}|}$ and the existence problem of a quasi-order $\succsim$ satisfying

LDD($\mathcal{R}$) is reduced to the existence problem of a suitable assignment on indeterminates $w_1, \ldots, w_{|\mathcal{R}|}$ which satisfies $[\![\text{LDD}]\!](\mathcal{R})$.

Definition 3.6. An arithmetic constraint $[\![\text{LDD}]\!](\mathcal{R})$ is defined as follows.

$$[\![\text{Ldd}]\!](i, j, x_1 \cdots x_n) = (\textstyle\bigwedge_{1 \le l \le n}((w_{x_l} < w_i) \vee (w_{x_l} < w_j))) \vee$$
$$\textstyle\bigvee_{1 \le k \le n} \left[(\bigwedge_{1 \le l < k}(w_{x_l} < w_i)) \wedge (w_{x_k} = w_j) \wedge (\bigwedge_{k < l \le n}((w_{x_l} < w_i) \vee (w_{x_l} < w_j)))\right]$$
$$[\![\text{LDD}]\!](\mathcal{R}) = \textstyle\bigwedge_{i,j \in \mathcal{R}} \bigwedge_{\langle s,t \rangle \in \text{CP}(i,j)} \left([\![\text{Ldd}]\!](i, j, \text{JL}(s,t)) \wedge [\![\text{Ldd}]\!](j, i, \text{JR}(s,t))\right)$$

Theorem 3.7. *A linear TRS $\mathcal{R}$ is confluent if $[\![\text{LDD}]\!](\mathcal{R})$ is satisfiable.* ∎

Since constrains $[\![\text{LDD}]\!](\mathcal{R})$ is a boolean combination of linear arithmetic formulas (every monomial contains only one variable), the satisfiability of $[\![\text{LDD}]\!](\mathcal{R})$ is efficiently checked by an external SMT (SAT modulo theories) solver where the linear arithmetic is employed for the underlying theory.

4. Rule-labelling heuristic capable of non-right-linear rules

As explained in [13], the rule-labelling heuristic is not applicable to non-linear TRSs, but by adding some additional information to labels, the technique can be extended to handle (possibly non-right-linear) left-linear TRSs. To explain this extension, let us replicate a situation from Example 20 of [13]. Suppose $i : \mathsf{f}(x) \to \mathsf{h}(x,x) \in \mathcal{R}$. To apply the decreasing diagrams criterion (Proposition 3.1), one has to impose the local decreasingness for peaks arising from nested overlaps of the same rewrite rule i such as:

$$\begin{tikzcd}
 & \mathsf{f}(C[\mathsf{f}(\mathsf{a})]) \arrow[ddl, "i"'] \arrow[dr, "i"] & \\
 & & \mathsf{h}(C[\mathsf{f}(\mathsf{a})], C[\mathsf{f}(\mathsf{a})]) \arrow[d, "i"] \\
\mathsf{f}(C[\mathsf{h}(\mathsf{a},\mathsf{a})]) \arrow[dr, "i"'] & & \mathsf{h}(C[\mathsf{h}(\mathsf{a},\mathsf{a})], C[\mathsf{f}(\mathsf{a})]) \arrow[dl, "i"] \\
 & \mathsf{h}(C[\mathsf{h}(\mathsf{a},\mathsf{a})], C[\mathsf{h}(\mathsf{a},\mathsf{a})]) &
\end{tikzcd}$$

This peak, however, is not locally decreasing as $\to_i \circ \to_i$ does not have the form $\xrightarrow{*}_{\prec i} \circ \xrightarrow{=}_{\simeq i} \circ \xrightarrow{*}_{\prec i,i}$. Hence the rule-labelling heuristic fails.

An idea to solve this situation is to extend the label i to $\langle m, i \rangle$ where m is the number of occurrences of f on the path from the redex to the root and use the lexicographic comparison (denoted by $\succsim_{\text{lex}}$) in which the first component is compared with the usual ordering $\ge$ on natural numbers [13]. Then we have labeled rewrite steps $\mathsf{f}(C[\mathsf{f}(\mathsf{a})]) \to_{\langle n+1,i \rangle} \mathsf{f}(C[\mathsf{h}(\mathsf{a},\mathsf{a})])$ and $\mathsf{h}(C[\mathsf{f}(\mathsf{a})], C[\mathsf{f}(\mathsf{a})]) \to_{\langle n,i \rangle} \mathsf{h}(C[\mathsf{h}(\mathsf{a},\mathsf{a})], C[\mathsf{f}(\mathsf{a})]) \to_{\langle n,i \rangle} \mathsf{h}(C[\mathsf{h}(\mathsf{a},\mathsf{a})], C[\mathsf{f}(\mathsf{a})])$, provided that the context $C[\,]$ has n-occurrences of f on the path from the hole to the root. Then, by $\langle n+1, i \rangle \succ_{\text{lex}} \langle n, i \rangle$, the local decreasingness of the peak is ensured.

Although it is not mentioned in [13], one should note that this extended heuristic does not work if there is a rewrite rule such as $j : \mathsf{g}(x) \to \mathsf{f}(x)$ in $\mathcal{R}$ whose rewrite step may increase the number of occurrences of f above a redex. For example, a critical peak below arising from the nested overlap of redex patterns is not locally decreasing:

$$\begin{tikzcd}
 & \mathsf{g}(C[s]) \arrow[dl, "{\langle n,k\rangle}"'] \arrow[dr, "{\langle n,j\rangle}"] & \\
\mathsf{g}(C[s']) \arrow[dr, "{\langle n,j\rangle}"'] & & \mathsf{f}(C[s]) \arrow[dl, "{\langle n+1,k\rangle}"] \\
 & \mathsf{f}(C[s']) &
\end{tikzcd}$$

as $\langle n+1,k\rangle \succ_{\text{lex}} \langle n,k\rangle$ and $\langle n+1,k\rangle \succ_{\text{lex}} \langle n,j\rangle$.

To avoid such a situation, one needs to impose an additional condition: for any contexts $C_l[\,], C_r[\,]$ and $x \in V$ such that $C_l[x] \to C_r[x] \in \mathcal{R}$,

- $\sharp_{\mathsf{f}} C_l[\,] \geq \sharp_{\mathsf{f}} C_r[\,]$ if x is linear in $C_r[x]$, and
- $\sharp_{\mathsf{f}} C_l[\,] > \sharp_{\mathsf{f}} C_r[\,]$ if x is not linear in $C_r[x]$

where $\sharp_{\mathsf{f}}\, C[\,]$ denotes the number of occurrences of the function symbol f along the path from the hole to the root in the context $C[\,]$.

5. A generalization of rule-labelling heuristic for left-linear TRSs

In this section, we extend the encoding presented in Sec. 3 to the rule-labelling heuristic for left-linear TRSs explained in Sec. 4 under the following natural generalization:

(1) Not only f but some subset $\mathcal{G}$ of function symbols can be designated for counting occurrences (on the path from the redex to the root).
(2) More generally, the counting of occurrences can be generalized to the summation of weights of ≥ 0 assigned for each function symbol's occurrences.

The summation of weights is formalized by a notion of the weight of context.

Definition 5.1. Let $C[\,]$ be a context and $w : \mathcal{F} \to \mathbb{N}$ be a function where $\mathbb{N}$ is the set of natural numbers. The *weight* $\sharp C[\,]$ of a context $C[\,]$ is defined as follows.

$$\sharp C[\,] = \begin{cases} 0 & \text{if } C[\,] = \Box \\ w(f) + \sharp\tilde{C}[\,] & \text{if } C[\,] = f(\ldots, \tilde{C}[\,], \ldots) \end{cases}$$

To encode the weight $\sharp C[\,]$, we introduce a non-negative integer variable z_f for each $f \in \mathcal{F}$ to be assigned by $w(f)$. Then $\sharp C[\,]$ is encoded by a polynomial $[\![\sharp C[\,]]\!]$ whose definition is obtained by replacing $w(f)$ by z_f in the definition of $\sharp C[\,]$. Thus the label of each rewrite step is encoded by $\langle \varphi, x\rangle$ where $x \in \{1, \ldots, |\mathcal{R}|\}$ and φ is a polynomial over indeterminate $(z_f)_{f\in\mathcal{F}}$. We assume that $\mathrm{JL}(s,t)$ and $\mathrm{JR}(s,t)$ are updated accordingly. The set $\mathrm{CP}_2(i,j)$ of critical pairs equipped with the weight of peak rewrite steps is given like this: $\mathrm{CP}_2(i,j)$ $= \{\big\langle\langle[\![\sharp l_j[\,]_p\sigma]\!], s\rangle, \langle 0, t\rangle\big\rangle \mid s = l_j[r_i]_p\sigma \leftarrow l_j[l_i]_p\sigma = l_j\sigma \to r_j\sigma = t, \langle s,t\rangle \in \mathrm{CP}(i,j)\}$.

Definition 5.2. Arithmetic constraints $[\![\mathrm{LDD}_2]\!](\mathcal{R})$ and $[\![\mathrm{CND}]\!](\mathcal{R})$ are defined as follows.

$$\langle\varphi,i\rangle \prec_{\text{lex}} \langle\rho,j\rangle = (\varphi < \rho \lor (\varphi = \rho \land w_i < w_j)) \qquad \langle\varphi,i\rangle \simeq_{\text{lex}} \langle\rho,j\rangle = (\varphi = \rho \land w_i < w_j)$$

$$\begin{aligned}
[\![\mathrm{Ldd}_2]\!](\vec{\varphi}, \vec{\psi}, \vec{\rho}_1 \cdots \vec{\rho}_n) = {} & (\textstyle\bigwedge_{1\leq l\leq n}((\vec{\rho}_l \prec_{\text{lex}} \vec{\varphi}) \lor (\vec{\rho}_l \prec_{\text{lex}} \vec{\psi}))) \lor \\
& \textstyle\bigvee_{1\leq k\leq n} \big[(\bigwedge_{1\leq l<k}(\vec{\rho}_l \prec_{\text{lex}} \vec{\varphi})) \land (\vec{\rho}_k \simeq_{\text{lex}} \vec{\psi}) \land (\bigwedge_{k<l\leq n}((\vec{\rho}_l \prec_{\text{lex}} \vec{\varphi}) \lor (\vec{\rho}_l \prec_{\text{lex}} \vec{\psi})))\big]
\end{aligned}$$

$$\begin{aligned}
[\![\mathrm{LDD}_2]\!](\mathcal{R}) = \textstyle\bigwedge_{i,j\in\mathcal{R}} \bigwedge_{\langle\langle\varphi,s\rangle,\langle\psi,t\rangle\rangle\in\mathrm{CP}_2(i,j)} \big(& [\![\mathrm{Ldd}_2]\!](\langle\varphi,i\rangle, \langle\psi,j\rangle, \mathrm{JL}(s,t)) \\
& \land [\![\mathrm{Ldd}_2]\!](\langle\psi,j\rangle, \langle\varphi,i\rangle, \mathrm{JR}(s,t))\big)
\end{aligned}$$

$$\begin{aligned}[\![\mathrm{CND}]\!](\mathcal{R}) = {} & \bigwedge\{[\![\sharp C_l[\,]]\!] \geq [\![\sharp C_r[\,]]\!] \mid C_l[x] \to C_r[x] \in \mathcal{R}, x \notin \mathrm{V}(C_r[\,])\} \\ & \wedge\ \bigwedge\{[\![\sharp C_l[\,]]\!] > [\![\sharp C_r[\,]]\!] \mid C_l[x] \to C_r[x] \in \mathcal{R}, x \in \mathrm{V}(C_r[\,])\}\end{aligned}$$

We here explain the constraint $[\![\mathrm{CND}]\!](\mathcal{R})$ by an example.

Example 5.3. Let $\mathcal{R} = \{\ \mathsf{f}(\mathsf{g}(x), y) \to \mathsf{h}(x, \mathsf{f}(x, y))\ \}$. Then the condition for variable y which is linear in the right-hand side (rhs for short) of the rule is encoded like this:

$$(1)\ z_\mathsf{f} \ \geq\ z_\mathsf{h} + z_\mathsf{f} \qquad (\text{for } C_l[\,] = \mathsf{f}(\mathsf{g}(x), \Box), C_r[\,] = \mathsf{h}(x, \mathsf{f}(x, \Box))).$$

The condition for variable x which is non-linear in rhs of the rule is encoded like this:

$$\begin{aligned}&(2)\ z_\mathsf{f} + z_\mathsf{g} \ >\ z_\mathsf{h} && (\text{for } C_l[\,] = \mathsf{f}(\mathsf{g}(\Box), y), C_r[\,] = \mathsf{h}(\Box, \mathsf{f}(x, y))),\\ &(3)\ z_\mathsf{f} + z_\mathsf{g} \ >\ z_\mathsf{h} + z_\mathsf{f} && (\text{for } C_l[\,] = \mathsf{f}(\mathsf{g}(\Box), y), C_r[\,] = \mathsf{h}(x, \mathsf{f}(\Box, y))).\end{aligned}$$

Therefore $[\![\mathrm{CND}]\!](\mathcal{R}) = (1) \wedge (2) \wedge (3)$.

Theorem 5.4. *A left-linear TRS $\mathcal{R}$ is confluent if $[\![\mathrm{LDD}_2]\!](\mathcal{R}) \wedge [\![\mathrm{CND}]\!](\mathcal{R})$ is satisfiable.* ■

Example 5.5. The following TRS $\mathcal{R}_1$ is from [4] (Example 20 of [14]).

$$\mathcal{R}_1 = \left\{ \begin{array}{llllllllllll} (1) & \mathsf{g}(\mathsf{a}) & \to & \mathsf{f}(\mathsf{g}(\mathsf{a})) & (3) & \mathsf{a} & \to & \mathsf{b} & (5) & \mathsf{h}(x,y) & \to & \mathsf{c} \\ (2) & \mathsf{g}(\mathsf{b}) & \to & \mathsf{c} & (4) & \mathsf{f}(x) & \to & \mathsf{h}(x,x) & & & & \end{array} \right\}$$

There is a (unique) critical peak of $\mathcal{R}$: $\mathsf{g}(\mathsf{b}) \leftarrow_{\langle z_\mathsf{g}, w_3\rangle} \mathsf{g}(\mathsf{a}) \to_{\langle 0, w_1\rangle} \mathsf{f}(\mathsf{g}(\mathsf{a}))$, which is joinable as $\mathsf{g}(\mathsf{b}) \to_{\langle 0, w_2\rangle} c \leftarrow_{\langle 0, w_5\rangle} \mathsf{h}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \leftarrow_{\langle 0, w_4\rangle} \mathsf{f}(\mathsf{g}(\mathsf{a}))$. By solving the constraint, a solution $z_\mathsf{f} = w_i = 1$ ($i \in \{1,3,5\}$), $z_f = w_2 = w_4 = 0$ ($f \in \{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{g},\mathsf{h}\}$) is obtained.

Example 5.6. Let $\mathcal{R}_1$ be the TRS given in Example 5.5 and $\mathcal{R}_2 = \mathcal{R}_1 \cup \{\mathsf{g}(x) \to \mathsf{f}(\mathsf{f}(x))\}$. The weight assignment in Example 5.5 does not work for $\mathcal{R}_2$ because of $[\![\mathrm{CND}]\!](\mathcal{R}_2)$. In fact, the generated constraint $[\![\mathrm{LDD}_2]\!](\mathcal{R}_2) \wedge [\![\mathrm{CND}]\!](\mathcal{R}_2)$ is not satisfiable if we limit $z_f \in \{0,1\}$. By solving the constraint, a solution $z_\mathsf{f} = w_i = 1$ ($i \in \{1,5,6\}$), $z_\mathsf{g} = w_2 = 2$, $w_3 = 3$, $z_f = w_5 = 0$ ($f \in \{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{g},\mathsf{h}\}$) is obtained. Thus one concludes $\mathcal{R}_2$ is confluent. This example demonstrates that our generalization from counting of function symbol's occurrences to summation of weight properly extends the applicability of the rule-labelling heuristic.

6. Adding further flexibilities to the rule-labelling heuristic

In this section, we add two further flexibilities to the rule-labelling heuristic.

6.1. Adding flexibility on the lexicographic comparison

In the previous section (and also in [13]) the label $\langle \sharp C[\,], i\rangle$ is compared in such a way that first on the weight $\sharp C[\,]$ and then on the weight of the rule i. It is easy to see, however, that comparing the components in the reverse order can be used either.

Example 6.1. Let

$$\mathcal{R}_3 = \left\{ \begin{array}{llllllllllll} (1) & \mathsf{c} & \to & \mathsf{f}(\mathsf{a}) & (3) & \mathsf{a} & \to & \mathsf{g}(\mathsf{a}) & (5) & \mathsf{f}(x) & \to & \mathsf{h}(x,x) \\ (2) & \mathsf{c} & \to & \mathsf{f}(\mathsf{b}) & (4) & \mathsf{b} & \to & \mathsf{g}(\mathsf{g}(\mathsf{a})) & & & & \end{array} \right\}.$$

First note that in $\mathcal{R}_3$ $z_\mathsf{f} > z_\mathsf{h}$ need to be satisfied by the rule (5). Consider the critical peak $\mathsf{f}(\mathsf{a}) \leftarrow_{\langle 0, w_1\rangle} \mathsf{c} \to_{\langle 0, w_2\rangle} \mathsf{f}(\mathsf{b})$ and a joinable rewrite sequence $\mathsf{f}(\mathsf{a}) \to_{\langle z_\mathsf{f}, w_3\rangle} \mathsf{f}(\mathsf{g}(\mathsf{a})) \to_{\langle z_\mathsf{f} + z_\mathsf{g}, w_3\rangle} \mathsf{f}(\mathsf{g}(\mathsf{g}(\mathsf{a}))) \leftarrow_{\langle z_\mathsf{f}, w_4\rangle} \mathsf{f}(\mathsf{b})$ for it. As z_f is positive, there is no chance to satisfy local decreasingness condition if the comparison by $\langle \sharp C[\,], i\rangle$ is used. On the other hand, if one uses the comparison by $\langle i, \sharp C[\,]\rangle$, a suitable assignment is found.

Another workaround here is to consider another joinable rewrite sequence with auxiliary (duplicating) rewrite steps by the rule (5): $\mathsf{f}(\mathsf{a}) \to \mathsf{h}(\mathsf{a},\mathsf{a}) \xrightarrow{*} \circ \xleftarrow{*} \mathsf{h}(\mathsf{b},\mathsf{b}) \leftarrow \mathsf{f}(\mathsf{b})$. For this, however, it is required to search a joinable rewrite sequence with non-minimal length.

Benefit from both ways of comparison is obtained easy—it suffices to prepare new integer variables $w'_1,\ldots,w'_{|\mathcal{R}|}$ and change the encoding $\langle w_i,\varphi\rangle$ to $\langle w_i,\varphi,w'_i\rangle$, where the third component is used to encode the secondary quasi-order on rules compared after the comparison of the context weight φ.

6.2. Adding flexibility on the weight function

It is also easy to see that the weight function $\sharp$ for the context can be changed in such a way that the counted weight on an occurrence of the same function symbol is changed according to the argument position containing the hole.

Definition 6.2. Let $C[\,]$ be a context and $w : \mathcal{F}_{\mathbb{N}} \to \mathbb{N}$ be a function, where $\mathcal{F}_{\mathbb{N}} = \{\langle f,i\rangle \mid f \in \mathcal{F}, 1 \le i \le \mathrm{arity}(f)\}$. The weight $\sharp' C[\,]$ of the context $C[\,]$ is defined as follows.

$$\sharp' C[\,] = \begin{cases} 0 & \text{if } C[\,] = \Box \\ w(f,i) + \sharp'\tilde{C}[\,] & \text{if } C[\,] = f(\ldots,\tilde{C}[],\ldots) \text{ and } C[\,]/i = \tilde{C}[\,] \end{cases}$$

To encode the weight $\sharp' C[\,]$, non-negative integer variables $(z_{f,i})_{\langle f,i\rangle\in\mathcal{F}_{\mathbb{N}}}$ are introduced. Using the weight function $\sharp'$ rather than $\sharp$ is sometimes advantageous as witnessed in the following example.

Example 6.3. Let

$$\mathcal{R}_4 = \left\{ \begin{array}{llcllll} (1) & \mathsf{f}(\mathsf{f}(x,y),z) & \to & \mathsf{f}(x,\mathsf{f}(y,z)) & (3) & \mathsf{f}(x,\mathsf{1}) \to & \mathsf{f}(\mathsf{1},x) \\ (2) & \mathsf{f}(\mathsf{1},x) & \to & x & & & \end{array} \right\}.$$

For a critical peak $\mathsf{f}(\mathsf{f}(w,\mathsf{f}(x,y)),z)) \leftarrow_{\langle z_{\mathsf{f},1},w_1\rangle} \mathsf{f}(\mathsf{f}(\mathsf{f}(w,x),y),z) \to_{\langle 0,w_1\rangle} \mathsf{f}(\mathsf{f}(w,x),\mathsf{f}(y,z)))$, there is a joinable rewrite sequence $\mathsf{f}(\mathsf{f}(w,\mathsf{f}(x,y)),z)) \to_{\langle 0,w_1\rangle} \mathsf{f}(w,\mathsf{f}(\mathsf{f}(x,y),z)) \to_{\langle z_{\mathsf{f},2},w_1\rangle} \mathsf{f}(w,\mathsf{f}(x,\mathsf{f}(y,z))) \leftarrow_{\langle 0,w_1\rangle} \mathsf{f}(\mathsf{f}(w,x),\mathsf{f}(y,z)))$. It is readily convinced that the diagram can not be made locally decreasing unless we distinguish $z_{\mathsf{f},1}$ and $z_{\mathsf{f},2}$.

7. Implementation and experiments

All techniques described in this paper have been implemented. The implementation is written in SML/NJ[1] and built upon the confluence prover ACP. We have used Yices[2] [3] as an external SMT solver. In searching of a joinable rewrite sequence of critical pairs, the following heuristics are employed: (i) set the maximum number of rewrite steps to 5. (ii) joinability is tested from reducts obtained in smaller steps (all joinable sequences obtained in the smallest step are considered but not any others with larger steps.)

We have tested various versions of rule-labelling heuristic described in this paper. The summary of experiments is described in Table 1. (1)–(5) are results of confluence proofs by decreasing diagrams based on the rule-labelling heuristics. We have also presented results of other confluence proving techniques for left-linear TRSs for comparison. The columns below the title $\mathcal{R}_i$ show success (✓) or failure (×) of the proof attempts to TRSs $\mathcal{R}_1$–$\mathcal{R}_4$

[1]`http://www.smlnj.org/`

[2]`http://yices.csl.sri.com/`

	$\mathcal{R}_1$	$\mathcal{R}_2$	$\mathcal{R}_3$	$\mathcal{R}_4$	*Col.*(msec)
Decreasing diagrams technique based on rule-labelling					
(1) basic version (Thm. 3.7)	×	×	×	×	35 (200)
(2) counting designated function symbol's occurrences	✓	×	×	×	41 (486)
(3) with context weight (Thm. 5.4)	✓	✓	×	×	41 (481)
(4) (3) + extended comparison (Subsect. 6.1)	✓	✓	✓	×	41 (795)
(5) (4) + extended context weight (Subsect. 6.2)	✓	✓	✓	✓	42 (692)
Other techniques for left-linear TRSs					
development closed TRSs [12]	×	×	×	×	16 (52)
linear strongly closed TRSs [6]	×	×	×	×	24 (52)
criterion by parallel critical pairs [9]	×	×	×	×	31 (58)
criterion by simultaneous critical pairs [7]	×	×	×	×	36 (91)
upside-parallel-closed/outside-closed TRSs [8]	×	×	×	×	19 (53)
All techniques	✓	✓	✓	✓	48 (593)
All techniques except the decreasing diagrams technique	×	×	×	×	40 (84)

Table 1: A summary of experiments

in the present paper. The columns below the title *Col.* show the number of success tested on a 106 collection of TRSs taken from various confluence-related papers and running time (msec.). All experiments have been performed on a FreeBSD platform of a PC equipped with 1.2GHz CPU and 1GB memory.

While other five techniques for left-linear TRSs proves 16–36 examples, the decreasing diagrams technique based on rule-labelling proves 45 examples ($\mathcal{R}_1$ is contained in the collection). Thus the comparison experimentally reveals the virtue of decreasing diagrams technique based on rule-labelling. The very basic version of the decreasing diagrams technique based on rule-labelling for linear TRSs already proves nearly 80% of the examples that can be proved with other extensions. Results on TRSs $\mathcal{R}_2$–$\mathcal{R}_4$ show that the refinements presented in the paper improve the applicability of the technique. The running time for decreasing diagrams technique based on rule-labelling is about 7–14 times larger than other five techniques. Since 34 examples are proved both in the decreasing diagrams technique based on rule-labelling and in the combination of other five techniques, it is better to try the other five techniques before the decreasing diagrams technique based on rule-labelling.

A new version of the confluence prover ACP involving all the techniques presented in the paper and the details of all experiments can be found on the webpage[3] of ACP.

8. Conclusion

We have described a method to automate confluence proofs by the decreasing diagrams based on the rule-labelling heuristic. We have shown an encoding of the confluence criterion into that of a linear arithmetic problem suitable for solving by external SMT solvers. An additional condition which need to be considered to guarantee the soundness of the technique (omitted in the original description of the heuristic [13]) and several generalizations of the heuristic which enlarge the applicability of the technique have been described. The presented technique has been implemented and the experiments show the advantage of incorporating the technique into automated confluence provers.

[3]`http://www.nue.riec.tohoku.ac.jp/tools/acp/`

Automation of decreasing diagram technique based on rule-labelling heuristic for linear TRSs has been obtained in [5] independently. Automation of the extended heuristic for left-linear TRSs, however, has not been explored in their paper. Instead, they are developing a new technique based on relative termination there.

In [13], another technique called self-duplication heuristic is described to deal with rule-labelling for (possibly non-right-linear) left-linear TRSs. In self-duplication heuristic, instead of counting function symbols' occurrences, parallel rewrite steps are considered to make critical peaks arising form nested overlaps of the non-right-linear rules locally decreasing. Automation of the decreasing diagrams technique with self-duplication heuristic remains as a future work.

Acknowledgments

The author thanks Yoshihito Toyama, Nao Hirokawa, Dominik Klein and anonymous referees for their helpful comments. This work was partially supported by a grant from JSPS No. 20500002.

References

[1] T. Aoto, Y. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In *Proc. of RTA 2009*, *LNCS*, vol. 5595, pp. 93–102. Springer-Verlag, 2009.

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

[3] B. Dutertre and L. de Moura. The YICES SMT solver. Available from `http://yices.csl.sri.com/tool-paper.pdf`.

[4] B. Gramlich and S. Lucas. Generalizing Newman's lemma for left-linear rewrite systems. In *Proc. of RTA 2006*, *LNCS*, vol. 4098, pp. 187–201. Springer-Verlag, 2006.

[5] N. Hirokawa and A. Middeldorp. Decreasing diagrams and relative termination. *Computing Research Repository*, 0910.2853, 2009. Unpublished manuscript.

[6] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

[7] S. Okui. Simultaneous critical pairs and Church-Rosser property. In *Proc. of RTA-98*, *LNCS*, vol. 1379, pp. 2–16. Springer-Verlag, 1998.

[8] M. Oyamaguchi and Y. Ohta. On the open problems concerning Church-Rosser of left-linear term rewriting systems. *IEICE Trans. Information and Systems*, E87-D(2):290–298, 2004.

[9] Y. Toyama. On the Church-Rosser property of term rewriting systems. Technical Report 17672, NTT ECL, 1981.

[10] Y. Toyama. Confluent term rewriting systems (invited talk). In *Proc. of RTA 2005*, *LNCS*, vol. 3467, p. 1. Springer-Verlag, 2005. Slides are available from `http://www.nue.riec.tohoku.ac.jp/user/toyama/slides/toyama-RTA05.pdf`.

[11] V. van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.

[12] V. van Oostrom. Developing developments. *Theoretical Computer Science*, 175(1):159–181, 1997.

[13] V. van Oostrom. Confluence by decreasing diagrams: converted. In *Proc. of RTA 2008*, *LNCS*, vol. 5117, pp. 306–320. Springer-Verlag, 2008.

[14] V. van Oostrom. Modularity of confluence: constructed. In *Proc. of IJCAR 2008*, *LNCS*, vol. 5195, pp. 348–363. Springer-Verlag, 2008.

[15] J. Yoshida, T. Aoto, and Y. Toyama. Automating confluence check of term rewriting systems. *Computer Software*, 26(2):76–92, 2009. In Japanese.