Confluence Analysis for Term Rewriting via Commutation

By Kiraku Shintani

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Nao Hirokawa

March, 2015

Confluence Analysis for Term Rewriting via Commutation

By Kiraku Shintani (1310032)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Nao Hirokawa

and approved by
Associate Professor Nao Hirokawa
Professor Mizuhito Ogawa
Professor Kazuhiro Ogata

February, 2015 (Submitted)

Contents

Chapter 1

Introduction

Term rewriting is a simple Turing complete computational model, which underlies automated theorem proving (e.g. E, Vampire, Waldmeister) and declarative programming languages (CafeOBJ, Haskell, OCaml). *Confluence* is a fundamental property that ensures uniqueness of computational results, which plays a crucial role in applications. While in programming languages confluence guarantees well-definedness of functions, in theorem proving confluence is used for equational reasoning.

Rewriting and Confluecne. In this thesis we investigate automated confluence analysis for term rewriting. A *term rewrite system* (TRS) is a directed equational system on terms. The following system $\mathcal{R}_2$ is an instance of left-linear TRSs:

$$\begin{array}{llll} 1\colon & \mathsf{eq}(\mathsf{a},\mathsf{a}) \to \mathsf{T} & 3\colon & \mathsf{eq}(\mathsf{a}*x,\mathsf{a}*y) \to \mathsf{eq}(x,y) \\ 2\colon & \mathsf{eq}(\mathsf{a},x*y) \to \mathsf{F} & 4\colon & \mathsf{eq}(x,y) \to \mathsf{eq}(y,x) \end{array} \qquad \begin{array}{ll} 5\colon & (x*y)*z \to x*(y*z) \\ 6\colon & x*(y*z) \to (x*y)*z \end{array}$$

The function eq checks if a given two sequences of a are identical. For instance, the term $\mathsf{eq}((\mathsf{a}*\mathsf{a})*\mathsf{a},\mathsf{a}*\mathsf{a})$ is computed by the following rewrite steps:

$$\begin{aligned} \mathsf{eq}((\mathsf{a}*\mathsf{a})*\mathsf{a},\mathsf{a}*\mathsf{a}) &\to_{\mathcal{R}} \mathsf{eq}(\mathsf{a}*(\mathsf{a}*\mathsf{a}),\mathsf{a}*\mathsf{a}) \\ &\to_{\mathcal{R}} \mathsf{eq}(\mathsf{a}*\mathsf{a},\mathsf{a}) \\ &\to_{\mathcal{R}} \mathsf{eq}(\mathsf{a},\mathsf{a}*\mathsf{a}) \\ &\to_{\mathcal{R}} \mathsf{F} \end{aligned}$$

The above is merely one way of computation. The term $\mathsf{eq}((\mathsf{a}*\mathsf{a})*\mathsf{a},\mathsf{a}*\mathsf{a})$ is also rewritten in a different way:

$$\begin{aligned} \mathsf{eq}((\mathsf{a}*\mathsf{a})*\mathsf{a},\mathsf{a}*\mathsf{a}) &\to_{\mathcal{R}} \mathsf{eq}(\mathsf{a}*\mathsf{a},(\mathsf{a}*\mathsf{a})*\mathsf{a}) \\ &\to_{\mathcal{R}} \mathsf{eq}(\mathsf{a}*\mathsf{a},\mathsf{a}*(\mathsf{a}*\mathsf{a})) \\ &\to_{\mathcal{R}} \mathsf{eq}(\mathsf{a},\mathsf{a}*\mathsf{a}) \\ &\to_{\mathcal{R}} \mathsf{F} \end{aligned}$$

Although the computation paths are different, the computational results are same. Here a natural question arises: Does such a uniqueness hold for any computation that starts from

the same term? Confluence addresses the issue. Since the above TRS fulfils confluence, uniqueness of results is guaranteed.

Automated Confluence Proving. Research of confluence has a long history, and many powerful confluence criteria have been proposed [7, 9, 12, 20, 21, 22]. Especially a significant amount of research exists for the class of *left-linear* TRSs, which model much of functional programs. Left-linearity means that for every rewrite rules each variable in the left-hand side occurs exactly once. In 2009 the first automatic confluence tool ACP [3] appeared. This triggered a renewed interest in renovating confluence criteria in the aspect of computability and efficiency [1, 2, 3, 6, 8, 11, 24], and development of confluence tools (CSI [24] and Saigawa).

We are concerned with associativity and/or commutativity rules and the commutation property of rewrite systems. Associativity and commutativity are fundamental notions in mathematics and computer science to axiomatize monoids and symmetric functions. The following TRS corresponds to a semi-group with a left-identity element:

$$1\colon \mathsf{0} + x \to x \qquad 2\colon x + (y + z) \to (x + y) + z \qquad 3\colon (x + y) + z \to x + (y + z)$$

None of confluence criteria [1, 2, 3, 6, 8, 9, 11, 12, 20, 22, 24] can show confluence of the TRS. Difficulty here is that standard techniques based on termination (e.g. [12, 8, 24]) and developments [9, 20, 22, 3] fail. Let's have a look at another TRS:

$$
\begin{array}{rcrc}
1\colon & \mathsf{0} \times y \to \mathsf{nil} & 9\colon & \mathsf{q}(\mathsf{q}(x,y),z) \to \mathsf{q}(x,\mathsf{q}(y,z)) \\
2\colon & \mathsf{s}(x) \times y \to y +\!\!+ (x \times y) & 10\colon & \mathsf{q}(x,\mathsf{q}(y,z)) \to \mathsf{q}(\mathsf{q}(x,y),z) \\
3\colon & \mathsf{hd}(\mathsf{c}(x)) \to x & 11\colon & \mathsf{q}(e,x) \to x \\
4\colon & \mathsf{hd}(\mathsf{c}(x) +\!\!+ y) \to x & 12\colon & \mathsf{q}(x,\mathsf{e}) \to x \\
5\colon & \mathsf{nil} +\!\!+ x \to x & 13\colon & x + (y + z) \to (x + y) + z \\
6\colon & x +\!\!+ \mathsf{nil} \to x & 14\colon & x + y \to y + x \\
7\colon & x +\!\!+ (y +\!\!+ z) \to (x +\!\!+ y) +\!\!+ z & 15\colon & \mathsf{0} + x \to x \\
8\colon & (x +\!\!+ y) +\!\!+ z \to x +\!\!+ (y +\!\!+ z) & 16\colon & \mathsf{s}(x) + y \to s(x + y) \\
 & & 17\colon & \mathsf{len}(\mathsf{e}) \to \mathsf{0} \\
 & & 18\colon & \mathsf{len}(\mathsf{a}) \to s(\mathsf{0}) \\
 & & 19\colon & \mathsf{len}(\mathsf{q}(\mathsf{a},y)) \to \mathsf{len}(\mathsf{a}) + \mathsf{len}(y)
\end{array}
$$

When analyzing such a complex TRS, *decomposition* is effective [3, 24]. In particular ACP showed powerfulness of commutation-based decomposition [20, 3]. According to the method, confluence of the TRS follows from confluence of the *commuting* subsystems (see Figure 1.1 for the commutation property): $\{1, 2, 5, 6, 7, 8\}$, $\{3, 4, 5, 6, 7, 8\}$, and $\{9, 10, \ldots, 19\}$. However, due to an exponential number of possible subsystems, the computational cost of the decomposition method is high.

Approach. In this thesis we propose a confluence analysis for left-linear TRSs *via commutation*. Commutation is a generalization of the confluence property (see Figure 1.1).

Figure 1.1: Confluence (left) and commutation (right)

The celebrated Commutation Theorem of Hindley [7] enables us to decompose the confluence problem of a complex TRS into a group of commutation problems of its subsystems. As direct methods for commutation, we employ confluence criteria including *rule labeling* [1] and the *Church-Rosser modulo* theorem [10], recasting them in commutation criteria. In order to derive the power of the Church-Rosser modulo theorem we have to perform equational unification, automation of which is one of the highlights of this thesis. In addition to those core contributions, we introduce several techniques useful for improving power and efficiency of confluence analysis. We remark that left-linearity of TRSs is an essential property of commutation, in fact many commutation criteria require left-linearity.

Overview. We introduce notions and notations in Chapter 2 that include the definition of TRS and rewrite steps. In Chapter 3 we introduce Jouannaud and Kirchner's criterion [10] for Church-Rosser modulo. We show unification algorithms and the coherence problem for automation of the criterion. In Chapter 4 we recall three existing commutation criteria and discuss a decomposition technique based on Hindley's commutation theorem and composability. All presented techniques have been implemented in our confluence tool CoLL. Chapter 5 reports experimental results based on the tool. Finally, in Chapter 6 we conclude the thesis with a discussion of related work and future directions.

Contributions. Here we list the main contributions of the thesis:

- a confluence proof by Church-Rosser modulo associativity and/or commutativity theories (Chapter 3),
- a commutation-based confluence analysis (Chapter 4),
- composability decomposition (Chapter 4),
- redundant rule elimination (Chapter 5),
- signature extension for commutation (Chapter 6), and
- the powerful confluence tool CoLL:

`http://www.jaist.ac.jp/project/saigawa/coll/`

Chapter 2

Preliminaries

We introduce notions and notations on term rewrite systems. Term rewriting is built on terms and relations on them. In this paper, we denote non-negative integers by $\mathbb{N}$. We assume basic knowledge of set theory.

2.1 Relations and Orders

Abstract notion of rewriting is formalized as binary relations.

Definition 2.1. *A (*binary*)* relation $\to$ *on a set A is a subset of $A \times A$. We denote $(a, b) \in \to$ by $a \to b$. A pair $\langle A, \to \rangle$ of a set A and a relation on A is called* abstract rewrite system (ARS)*. We denote the relation $\to$ of an ARS $\mathcal{A}$ by $\to_{\mathcal{A}}$. Throughout the thesis, we use relations and ARSs interchangeably.*

The next definition introduces several important operations on relations. The notation $\to_1 \cdot \to_2$ stands the *composition* of relations $\to_1$ and $\to_2$ defined by $\to_1 \cdot \to_2 = \{ (a, c) \mid a \to_1 b \text{ and } b \to_2 c \}$.

Definition 2.2. *Let $\to$ be a relation on a set A. The compositions of $\to$ are defined as follows:*

- *The n-*step relation *is defined for all $n \in \mathbb{N}$ as follows:*

$$\to^n = \begin{cases} \{ (x, x) \mid x \in A \} & \text{if } n = 0 \\ \to \cdot \to^{n-1} & \text{if } n > 0 \end{cases}$$

- *The* reflexive closure $\to^=$ *is defined as $\to \cup \to^0$.*
- *The* transitive closure $\to^+$ *is defined as $\bigcup_{i>0} \to^i$.*
- *The* reflexive transitive closure $\to^*$ *is defined as $\bigcup_{i\geqslant 0} \to^i$.*
- *The* symmetric closure $\leftrightarrow$ *is defined as $\to \cup \leftarrow$.*

- *The* inverse $\leftarrow$ *is defined as* $\{\ (b,a) \mid a \to b\ \}$.
- *The relation* $\leftarrow \cdot \rightarrow$ *is a* peak.
- *The relation* $\rightarrow \cdot \leftarrow$ *is a* valley.
- *Two elements* $a, b \in A$ *are* joinable *if* $a \to^* \cdot {}^*\!\leftarrow b$ *holds.*

Definition 2.3. *Let* $\to$ *be a relation on* A. *An element* $a \in A$ *is a* normal form *if there is no* $b \in A$ *such that* $a \to b$. *The set of all normal forms of* $\to$ *is denoted by* $\mathsf{NF}(\to)$.

Definition 2.4. *A relation* $\to$ *is* well-founded *if there are no infinite sequence such that*

$$a_1 \to a_2 \to a_3 \to \cdots$$

A relation is terminating *if it is well-founded.*

Commutation is a property on interaction of two relations.

Definition 2.5. *Let* $\to_1$ *and* $\to_2$ *be relations.*

- *The relations* commute *if* ${}^*_1\!\leftarrow \cdot \to_2^* \subseteq \to_2^* \cdot {}^*_1\!\leftarrow$ *holds.*
- *The relations* locally commute *if* ${}_1\!\leftarrow \cdot \to_2 \subseteq \to_2^* \cdot {}^*_1\!\leftarrow$ *holds.*

A relation is called confluent *if the relation self-commutes, i.e., the relation commutes with itself.*

With the next example we explain the confluence property.

Example 2.6. *Let* $\mathcal{A}_1$ *be the ARS* $\langle \{a,b,c\}, \{a \to b, a \to c, b \to c\} \rangle$. *Since every peak of* $\mathcal{A}_1$ *is joinable at the element* c, *the ARS* $\mathcal{A}_1$ *is confluent. Consider the another ARS* $\mathcal{A}_2 = \langle \{a,b,c\}, \{a \to b, a \to c\} \rangle$. *Because the peak* $b\ {}_{\mathcal{A}_2}\!\leftarrow a \to_{\mathcal{A}_2} c$ *is not joinable,* $\mathcal{A}_2$ *is not confluent.*

We define several types of orders. We say that an irreflexive and transitive relation $>$ is a *strict order*, and a reflexive and transitive relation $\geqslant$ is a *preorder*.

Definition 2.7. *Let* $>$ *be a strict order. The* lexicographic order $>_{lex}$ *is defined as follows:*

$$(a_1, b_1) >_{lex} (a_2, b_2) \iff a_1 > a_2 \text{ or, } a_1 = a_2 \text{ and } b_1 > b_2$$

Definition 2.8. *Let* A *be a set. A* multiset M *over* A *is a function* A *to* $\mathbb{N}$. *Suppose* M *and* N *are multisets over* A. *The basic multiset operators are defined as follows:*

- *The* $x \in M$ *holds if* $M(x) > 0$.
- *The* $M \subseteq N$ *holds if* $M(x) \leqslant N(x)$ *for all* $x \in A$.
- *The union* $M \cup N$ *is the function defined by* $(M \cup N)(x) = M(x) + N(x)$.

- *The difference $M \setminus N$ is defined as follows:*

$$(M \setminus N)(x) = \begin{cases} 0 & \text{if } M(x) < N(x) \\ M(x) - N(x) & \text{if } M(x) \geqslant N(x) \end{cases}$$

Definition 2.9. *Let M and N be multisets over A and $>$ strict order on A. The* multiset extension $M >_{\mathrm{mul}} N$ *holds if there is a set $X \subseteq M$ such that $X \neq \varnothing$, $N \supseteq (M \setminus X)$ and $x > y$ for some $x \in X$ for all $y \in N \setminus (M \setminus X)$. We write $M \geqslant_{\mathrm{mul}} N$ if $M >_{\mathrm{mul}} N$ or $M = N$.*

2.2 Terms and Substitutions

Definition 2.10. *A* signature $\mathcal{F}$ *is a set of function symbols, where each function symbol f is associated with non-negative integer n, the* arity *of f. We denote the arity of a function symbol f by $f^{(n)}$. If $c \in \mathcal{F}$ has 0-arity, we call c a* constant.

Terms are defined both from a signature $\mathcal{F}$ and a set $\mathcal{V}$ of variables.

Definition 2.11. *Let $\mathcal{F}$ be a signature and $\mathcal{V}$ a set of variables with $\mathcal{F} \cap \mathcal{V} = \varnothing$. The set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of* terms *is defined as follows:*

- $x \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ *if $x \in \mathcal{V}$, and*
- $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ *if $f^{(n)} \in \mathcal{F}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.*

We use small letters $s, t, u, \ldots$ for terms and $x, y, z, \ldots$ for variables.

Definition 2.12. *Let t be a term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$.*

- *The set $\mathcal{F}\mathsf{un}(t)$ of all function symbols in t is denoted as:*

$$\mathcal{F}\mathsf{un}(t) = \begin{cases} \varnothing & \text{if } t \in \mathcal{V} \\ \{f\} \cup \bigcup_{i=1}^{n} \mathcal{F}\mathsf{un}(t_i) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

- *The set $\mathcal{V}\mathsf{ar}(t)$ of all variables in t is denoted as:*

$$\mathcal{V}\mathsf{ar}(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V} \\ \bigcup_{i=1}^{n} \mathcal{V}\mathsf{ar}(t_i) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

- *The set $\mathcal{P}\mathsf{os}(t)$ of finite sequences of positive numbers, called* positions*, is defined as follows:*

$$\mathcal{P}\mathsf{os}(t) = \begin{cases} \epsilon & \text{if } t \in \mathcal{V} \\ \epsilon \cup \{\, ip \mid 1 \leqslant i \leqslant n, p \in \mathcal{P}\mathsf{os}(t_i) \,\} & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

  where ϵ is empty sequence named root *position.*

- *The set* $\mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$ *of positions corresponded non variable subterms is defined as:*

$$\mathcal{P}\mathsf{os}_{\mathcal{F}}(t) = \begin{cases} \varnothing & \text{if } t \in \mathcal{V} \\ \epsilon \cup \{\, ip \mid 1 \leqslant i \leqslant n, p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(t_i) \,\} & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

- *The root symbol* $\mathrm{root}(t)$ *of* t *is defined as follows:*

$$\mathrm{root}(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Definition 2.13. *The notation* $s|_p$ *denotes the subterm of the term* s *at the position* $p \in \mathcal{P}\mathsf{os}(s)$*, i.e.,*

$$s|_p = \begin{cases} s & \text{if } p = \epsilon \\ t_i|_q & \text{if } p = iq \text{ and } s = f(t_1, \ldots, t_i, \ldots, t_n) \end{cases}$$

The notation $s[t]_p$ *denotes the term resulting from replacing the subterm at the position* $p \in \mathcal{P}\mathsf{os}(s)$ *by* t*:*

$$s[t]_p = \begin{cases} t & \text{if } p = \epsilon \\ f(s_1, \ldots, s_i[t]_q, \ldots, s_n) & \text{if } p = iq \text{ and } s = f(s_1, \ldots, s_i, \ldots, s_n) \end{cases}$$

Definition 2.14. *A term* s *is a* subterm *of* t *if there is a position* $p \in \mathcal{P}\mathsf{os}(t)$ *such that* $t|_p = s$*. If* $p \neq \epsilon$ *then* s *is a* proper subterm *of* t*. We write* $s \trianglelefteq t$ *(*$s \triangleleft t$*) if* s *is a (proper) subterm of* t*.*

Definition 2.15. *Let* $\Box$ *be a fresh constant called the* hole*. A term* $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$ *is a* context *if* C *contains exactly one hole* $\Box$*. The term* $C[t]$ *is inductively defined as follows:*

$$C[t] = \begin{cases} t & \text{if } C = \Box \\ f(t_1, \ldots, C'[t], \ldots, t_n) & \text{if } C = f(t_1, \ldots, C', \ldots, t_n) \text{ with context } C' \end{cases}$$

Definition 2.16. *Let* $\mathcal{V}$ *be a set of variables and* t *in* $\mathcal{T}(\mathcal{F}, \mathcal{V})$*. An assignment* σ *on* $\mathcal{V}$ *to* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *is called* substitution *over* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *with the property that its domain* $\mathcal{D}\mathsf{om}(\sigma)$ *is finite.*

- *We write* $t\sigma$ *for a term replaced variables by a substitution* σ *such that*

$$t\sigma = \begin{cases} \sigma(t) & \text{if } t \in \mathcal{V} \\ f(t_1\sigma, \ldots, t_n\sigma) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

- *A set* $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ *of* variable bindings *represents a substitution* σ *such that* $\sigma(x_i) = t_i$ *for all* $1 \leqslant i \leqslant n$*.*

- *The application $t\sigma\tau$ of a composited substitution $\sigma\tau$ is defined by $(t\sigma)\tau$.*

Definition 2.17. *A bijective substitution is called a* renaming. *The terms s and t are* variant *if there is a renaming σ such that $s\sigma = t$. The pairs (s_1, t_1) and (s_2, s_2) are variant if there is a renaming σ such that $s_1\sigma = s_2$ and $t_1\sigma = t_2$.*

The next definition is a main topic of this section.

Definition 2.18. *A* unifier *σ of s and t is substitution such that $s\sigma = t\sigma$. A unifier σ of s and t is a* most general unifier *if for every unifier σ' of s and t there is some substitution τ such that $\sigma\tau = \sigma'$.*

An equality $s \approx t$ is a pair of terms s and t. We use the notation to express unification problems.

Example 2.19. *We show few examples of unification on $\mathcal{T}(\{\mathsf{f}, +, \mathsf{a}, \mathsf{b}\}, \mathcal{V})$. Consider the following equation:*

$$x \approx \mathsf{f}(\mathsf{a})$$

The equation has exactly one unifier $\{x \mapsto \mathsf{f}(\mathsf{a})\}$. The following each equations are not unifiable:

$$x \approx \mathsf{f}(x) \qquad \mathsf{a} + x \approx \mathsf{b} + y$$

In the former equation the variable x appears in the different terms in both sides, while in the latter equation the left-hand-side of the terms are different. Therefore they admit no unifier. Consider the following equation:

$$x \approx \mathsf{f}(y)$$

Whereas there are infinitely many unifiers

$$\{x \mapsto \mathsf{f}(y)\}, \{x \mapsto \mathsf{f}(\mathsf{a}), y \mapsto \mathsf{a}\}, \{x \mapsto \mathsf{f}(\mathsf{f}(y)), y \mapsto \mathsf{f}(y)\}, \cdots$$

the most general unifier $\{x \mapsto \mathsf{f}(y)\}$ subsumes all unifiers.

We extend unification so as to make terms identical in a given equational theory.

Definition 2.20. *Let $\mathcal{E}$ be an equational theory. An* $\mathcal{E}$-unifier *σ of s and t is substitution such that $s\sigma \leftrightarrow^*_{\mathcal{E}} t\sigma$ for some substitution σ. A set $\mathcal{U}$ of $\mathcal{E}$-unifiers of s and t is complete if for every $\mathcal{E}$-unifier σ' of s and t there is σ in $\mathcal{U}$ such that $\sigma\tau \leftrightarrow^*_{\mathcal{E}} \sigma'$ for some substitution τ. A minimal complete set $\mathcal{U}$ of $\mathcal{E}$-unifiers is a complete set satisfies the additional condition: for all $\sigma, \sigma' \in U$, if $\sigma\tau \leftrightarrow^*_{\mathcal{E}} \sigma'$ for some substitution τ then $\sigma = \sigma'$.*

We denote by $\mathcal{U}_{\mathcal{E}}(s \approx t)$ a fixed complete set of $\mathcal{E}$-unifiers for terms s and t.

Example 2.21. *We continue Example 2.19. Let $\to_{\mathcal{E}}$ be a relation such that $\to_{\mathcal{E}} = \{(\mathsf{a} + \mathsf{b}, \mathsf{b} + \mathsf{a})\}$. Consider the following equation that is not unifiable in syntactic unification:*

$$\mathsf{a} + x \approx \mathsf{b} + y$$

The equation is unifiable by the $\mathcal{E}$-unifier $\{x \mapsto \mathsf{b}, y \mapsto \mathsf{a}\}$ as $\mathsf{a} + \mathsf{b} \to^*_{\mathcal{E}} \mathsf{b} + \mathsf{a}$.

2.3 Term Rewrite Systems

Definition 2.22. *A pair of terms ℓ and r is called* rewrite rule *if ℓ is not variable and $\mathcal{V}\mathsf{ar}(\ell) \supseteq \mathcal{V}\mathsf{ar}(r)$. We write a rewrite rule (ℓ, r) by $\ell \to r$. A* term rewrite system *(TRS) is a set of rewrite rules.*

Usually we denote TRSs by $\mathcal{R}$ and $\mathcal{S}$. Consider the following TRS $\mathcal{R}_0$:

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

We use the TRS as a running example to explain notions introduced in this section.

Definition 2.23. *Let $\mathcal{R}$ be a TRS. The* rewrite step *$s \to_{\mathcal{R}} t$ is defined by there is rewrite rule $\ell \to r$ in $\mathcal{R}$ such that $s|_p = \ell\sigma$ and $t = s[r\sigma]p$ for some position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ and substitution σ.*

Given a TRS $\mathcal{R}$, we denote the set of normal forms of $\to_{\mathcal{R}}$ by $\mathsf{NF}(\mathcal{R})$. The term $\mathsf{s}(\mathsf{0}) + \mathsf{s}(\mathsf{0})$ is rewritten as follows:

$$\begin{aligned} \mathsf{s}(\mathsf{0}) + \mathsf{s}(\mathsf{0}) &\to_{\mathcal{R}_0} \mathsf{s}(\mathsf{0} + \mathsf{s}(\mathsf{0})) && \text{(by rule 2 with } p = \epsilon \text{ and } \sigma = \{x \mapsto \mathsf{0}, y \mapsto \mathsf{s}(\mathsf{0})\}) \\ &\to_{\mathcal{R}_0} \mathsf{s}(\mathsf{s}(\mathsf{0})) && \text{(by rule 1 with } p = 1 \text{ and } \sigma = \{x \mapsto \mathsf{s}(\mathsf{0})\}) \end{aligned}$$

The next definitions are used in remaining chapters.

Definition 2.24. *Let $\mathcal{R}$ be a TRS. The* multi-step *$\multimap_{\mathcal{R}}$ is inductively defined on terms as follows:*

1. *$x \multimap_{\mathcal{R}} x$ for all $x \in \mathcal{V}$,*
2. *$f(s_1, \ldots, s_n) \multimap_{\mathcal{R}} f(t_1, \ldots, t_n)$ if $s_i \multimap_{\mathcal{R}} t_i$ for all $1 \leqslant i \leqslant n$, and*
3. *$\ell\sigma \multimap_{\mathcal{R}} r\tau$ if $\ell \to r \in \mathcal{R}$ and $x\sigma \multimap_{\mathcal{R}} x\tau$ for all variables x.*

Definition 2.25. *Linearity is defined on terms, rewrite rules and TRSs:*

- *A term t is* linear *if in t every variables occurs exactly once.*
- *A rewrite rule $\ell \to r$ is* left-linear *if ℓ is linear.*
- *A TRS $\mathcal{R}$ is* left-linear *if every rewrite rule in $\mathcal{R}$ is left-linear.*

2.4 Confluence Analysis

Conditions for confluence are often based on the notion of *critical pairs*.

Definition 2.26. *Let $\ell_1 \to r_1$ be a rule in a TRS $\mathcal{R}$ and $\ell_2 \to r_2$ a variant of a rule in a TRS $\mathcal{S}$ with $\mathcal{V}\mathsf{ar}(\ell_1) \cap \mathcal{V}\mathsf{ar}(\ell_2) = \varnothing$. When $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$ and $\sigma \in \mathcal{U}_{\mathcal{E}}(\ell_1 \approx \ell_2|_p)$, the pair $(\ell_2\sigma[r_1\sigma]_p, r_2\sigma)$ is called an* $\mathcal{E}$-extended critical pair *(or simply* $\mathcal{E}$-critical pair*) of $\mathcal{R}$ on $\mathcal{S}$, and written $\ell_2\sigma[r_1\sigma]_p \;{}_{\mathcal{R},\mathcal{E}}\!\leftarrow\!\rtimes\!\rightarrow_{\mathcal{S}} r_2\sigma$.*

For $\mathcal{E} = \varnothing$ the set of all $\mathcal{E}$-critical pairs is denoted by ${}_{\mathcal{R}}\!\leftarrow\!\rtimes\!\rightarrow_{\mathcal{S}}$, and they are simply called critical pairs. For $\mathcal{R}_0$ there are two critical pairs:

$$\begin{array}{lll} \mathsf{s}(y') & {}_{\mathcal{R}_0}\!\leftarrow\!\rtimes\!\rightarrow_{\mathcal{R}_0} & \mathsf{s}(\mathsf{0}+y') \\ \mathsf{s}(x'+\mathsf{s}(y')) & {}_{\mathcal{R}_0}\!\leftarrow\!\rtimes\!\rightarrow_{\mathcal{R}_0} & \mathsf{s}(\mathsf{s}(x')+y') \end{array}$$

Consider the rules $\{1,2\}$ and $\mathsf{C} = \{x+y \to y+x\}$. Since $\mathsf{s}(x')+\mathsf{0} \leftrightarrow^*_{\mathsf{C}} \mathsf{0}+\mathsf{s}(x)$, we have C-critical pair $\mathsf{s}(\mathsf{0})\ {}_{\{1,2\},\mathsf{C}}\!\leftarrow\!\rtimes\!\rightarrow_{\{1,2\}}\ \mathsf{s}(x'+\mathsf{0})$.

Observe that termination of $\mathcal{R}_0$ is trivial. Under the termination assumption, confluence is characterized by the local confluence property, according to famous Newman's Lemma.

Theorem 2.27. *A terminating relation is confluent if and only if it is locally confluent.*

Local confluence is shown by the following Critical Pair Lemma [4].

Lemma 2.28. *A TRS is locally confluent if and only if all its critical pairs are joinable.* □

Since all critical pairs of $\mathcal{R}_0$ are joinable by ${}_{\mathcal{R}_0}\!\leftarrow$, confluence of $\mathcal{R}_0$ is concluded.

2.5 Church-Rosser modulo

Rewriting modulo $\mathcal{E}$ uses $\mathcal{E}$-pattern matching in rewriting, and the counterpart of the confluence property is called Church-Rosser modulo.

Definition 2.29. *Let $\mathcal{R}$ and $\mathcal{E}$ be TRSs. The rewrite step of $\mathcal{R}$* modulo theory *$\mathcal{E}$, denoted by $s \to_{\mathcal{R},\mathcal{E}} t$, is defined as follows: If $s|_p \leftrightarrow^*_{\mathcal{E}} \ell\sigma$ and $t = s[r\sigma]_p$ for some position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$, rule $\ell \to r \in \mathcal{R}$, and substitution σ.*

Definition 2.30. *The relation $\to_{\mathcal{R},\mathcal{E}}$ is* Church-Rosser modulo *$\mathcal{E}$, denoted as $\mathsf{CR}(\mathcal{R},\mathcal{E})$ if $\leftrightarrow^*_{\mathcal{R}\cup\mathcal{E}} \subseteq \to^*_{\mathcal{R},\mathcal{E}} \cdot \leftrightarrow^*_{\mathcal{E}} \cdot\ {}^*_{\mathcal{R},\mathcal{E}}\!\leftarrow$.*

Definition 2.31. *Let $\mathcal{F}_{\mathsf{A}}$, $\mathcal{F}_{\mathsf{C}}$ and $\mathcal{F}_{\mathsf{AC}}$ be pairwise disjoint sets of binary function symbols. We define the three theories A (associativity, A), C (commutativity, C), and AC as:*

$$\begin{aligned} \mathsf{A} &= \{\ f(f(x,y),z) \to f(x,f(y,z)),\ \ f(x,f(y,z)) \to f(f(x,y),z) \mid f \in \mathcal{F}_{\mathsf{A}}\ \} \\ \mathsf{C} &= \{\ f(x,y) \to f(y,x) \mid f \in \mathcal{F}_{\mathsf{C}}\ \} \\ \mathsf{AC} &= \{\ f(f(x,y),z) \to f(x,f(y,z)),\ \ f(x,y) \to f(y,x) \mid f \in \mathcal{F}_{\mathsf{AC}}\ \} \end{aligned}$$

In the next chapter we explain how to use Church-Rosser modulo in confluence analysis.

Chapter 3

Confluence via Church-Rosser Modulo

In this chapter we explain how the next theorem by Jouannaud and Kirchner [10] is used for confluence analysis. Especially, we discuss how to deal with associativity and/or commutativity rules.

Theorem 3.1. *Let $\mathcal{R}$ and $\mathcal{E}$ be TRSs that $\mathcal{R}/\mathcal{E}$ is terminating and $\rhd \cdot \leftrightarrow_{\mathcal{E}}^*$ is well-founded. Then, $\mathsf{CR}(\mathcal{R},\mathcal{E})$ if and only if ${}_{\mathcal{R},\mathcal{E}}\!\leftarrow \rtimes \rightarrow_{\mathcal{R}\cup\mathcal{E}\cup\mathcal{E}^{-1}} \subseteq \rightarrow_{\mathcal{R},\mathcal{E}}^* \cdot \leftrightarrow_{\mathcal{E}}^* \cdot {}_{\mathcal{R},\mathcal{E}}^*\!\leftarrow$.* □

We use the next left-linear TRS $\mathcal{R}_1$ to illustrate problems that arise when employing Theorem 3.1:

$$1\colon \mathsf{0} + x \to x \qquad 2\colon x + (y + z) \to (x + y) + z \qquad 3\colon (x + y) + z \to x + (y + z)$$

An idea here is proving $\mathsf{CR}(\{1\},\{2,3\})$ to conclude confluence of $\mathcal{R}_1$. The next trivial lemma validates this idea. We call a TRS $\mathcal{E}$ *reversible* if $\rightarrow_{\mathcal{E}} \subseteq {}_{\mathcal{E}}^*\!\leftarrow$ holds.

Lemma 3.2. *Suppose $\mathcal{E}$ is reversible. If $\mathsf{CR}(\mathcal{R},\mathcal{E})$ then $\mathcal{R}\cup\mathcal{E}$ is confluent.* □

Reversibility of $\{2,3\}$ and well-foundedness of $\rhd \cdot \leftrightarrow_{\{2,3\}}^*$ are trivial. Termination of $\{1\}/\{2,3\}$ can be shown by AC-RPO [17]. Therefore, it remains to test joinability of extended critical pairs to apply Theorem 3.1.

3.1 Associative Unification

How to compute A-critical pairs? Plotkin [15] introduced a procedure that enumerates a minimal complete set of A-unifiers. It is well-known that a minimal complete set need not to be finite, and thus the procedure may not terminate. In fact there is a TRS that admits infinitely many A-critical pairs. Probably this is one of main reasons that existing confluence tools do not support Theorem 3.1 for associativity rules. However, as observed in [18], a minimal complete set resulting from the procedure is finite whenever an input equality is a pair of linear terms that share no variables. Therefore, for every left-linear TRS one can safely use Plotkin's procedure to compute their A-critical pairs.

We present a simple A-unification procedure obtained by specializing Plotkin's procedure to our setting. Let S and T be sets of substitutions. We abbreviate the set $\{\sigma\tau \mid \sigma \in S \text{ and } \tau \in T\}$ to ST. Given a term t, we write $t{\downarrow_{\mathsf{A}'}}$ for the normal form of t with respect to A'. Here A' is the complete TRS $\{f(f(x,y),z) \to f(x,f(y,z)) \mid f \in \mathcal{F}_{\mathsf{A}}\}$.

Definition 3.3. *Let s and t be terms. The function $\langle s \approx t\rangle$ is inductively defined as follows.*

$$\langle s \approx t\rangle = \begin{cases} \{\{s \mapsto t\}\} & \text{if } s \in \mathcal{V} \\ \{\{t \mapsto s\}\} & \text{if } s \notin \mathcal{V} \text{ and } t \in \mathcal{V} \\ (A_1 \cdots A_n) \cup A_{s,t} \cup A_{t,s} & \text{if } s = f(s_1,\ldots,s_n) \text{ and } t = f(t_1,\ldots,t_n) \\ \varnothing & \text{otherwise} \end{cases}$$

where,

$$A_i = \langle s_i \approx t_i\rangle, \qquad A_{s,t} = \begin{cases} \{\{s_1 \mapsto f(t_1,s_1)\}\}\langle s \approx t_2\rangle & \text{if } (*) \\ \varnothing & \text{otherwise} \end{cases}$$

and $(*)$ stands for $s = f(s_1,s_2)$, $t = f(t_1,t_2)$, $f \in \mathcal{F}_{\mathsf{A}}$, and $s_1 \in \mathcal{V}$.

Lemma 3.4. *Let s and t be linear terms with $\mathcal{V}\mathsf{ar}(s) \cap \mathcal{V}\mathsf{ar}(t) = \varnothing$. If $\mu \in \langle s \approx t\rangle$ then $s\mu \leftrightarrow_{\mathsf{A}}^* t\mu$ and $\mathcal{D}\mathsf{om}(\mu) \subseteq \mathcal{V}\mathsf{ar}(s) \cup \mathcal{V}\mathsf{ar}(t)$.*

Proof. We perform induction on the multiset $\{s,t\}$ with respect to $\rhd^{\mathrm{mul}}$.

- If $s \in \mathcal{V}$ then $s\mu = t\mu$ and $\mathcal{D}\mathsf{om}(\mu) = \{s\} \subseteq \mathcal{V}\mathsf{ar}(s) \cup \mathcal{V}\mathsf{ar}(t)$. Similarly, the case of $s \notin \mathcal{V}$ and $t \in \mathcal{V}$ is also proved.
- If $s = f(s_1,\ldots,s_n)$ and $t = f(t_1,\ldots,t_n)$ with $f \notin \mathcal{F}_{\mathsf{A}}$ then we define μ_i as $\{x \mapsto x\mu \mid x \in \mathcal{V}\mathsf{ar}(s_i) \cup \mathcal{V}\mathsf{ar}(t_i)\}$. Due to linearity of s and t and $\mathcal{V}\mathsf{ar}(s) \cap \mathcal{V}\mathsf{ar}(t) = \varnothing$, we have $\mu = \mu_1 \cdots \mu_n$ and $\mu_i \in \langle s_i \approx t_i\rangle$. Thus, the induction hypotheses yield $s_i\mu_i \leftrightarrow_{\mathsf{A}}^* t_i\mu_i$ and $\mathcal{D}\mathsf{om}(\mu_i) \subseteq \mathcal{V}\mathsf{ar}(s_i) \cup \mathcal{V}\mathsf{ar}(t_i)$ for all i, and therefore, the claim follows.
- Otherwise, μ is derived from $A_{s,t} \cup A_{t,s}$ in the case of $s = f(s_1,s_2)$, $t = f(t_1,t_2)$ with $f \in \mathcal{F}_{\mathsf{A}}$. Without loss of generality assume $\mu \in A_{s,t}$. There is a substitution $\mu_2 \in \langle s \approx t_2\rangle_{\mathsf{A}}$ such that $\mathcal{D}\mathsf{om}(\mu_2) \subseteq \{s\} \cup \mathcal{V}\mathsf{ar}(t_1)$ and $\mu = \{s_1 \mapsto f(t_1,s_1)\}\mu_2$. By the assumption we have $\mathcal{V}\mathsf{ar}(s_1) \cap \mathcal{V}\mathsf{ar}(t) = \varnothing$. Therefore, by the induction hypothesis

  $$s\mu = f(f(t_1\mu_2, s_1\mu_2), s_2\mu_2) \to_{\mathsf{A}} f(t_1\mu_2, s\mu_2) \leftrightarrow_{\mathsf{A}}^* f(t_1\mu_2, t_2\mu_2) = t\mu$$

  is obtained. Moreover, we have $\mathcal{D}\mathsf{om}(\mu) = \{s_1\} \cup \mathcal{D}\mathsf{om}(\mu_2) \subseteq \mathcal{V}\mathsf{ar}(s) \cup \mathcal{V}\mathsf{ar}(t)$.

□

Let $\mathsf{A}' = \{f(f(x,y),z) \to f(x,f(y,z)) \mid f \in \mathcal{F}_{\mathsf{A}}\}$.

Lemma 3.5. *Suppose* $s, t \in \mathsf{NF}(\mathsf{A}')$ *and* $\mathcal{V}\mathsf{ar}(s) \cap \mathcal{V}\mathsf{ar}(t) = \varnothing$. *If* $s\sigma \leftrightarrow_{\mathsf{A}}^* t\sigma$ *then* $\mu \in \langle s \approx t \rangle$ *and* $\sigma \leftrightarrow_{\mathsf{A}}^* \mu\tau$ *for some substitutions* μ *and* τ.

Proof. By induction on the multiset $\{s, t\}$ with respect to $\rhd^{\mathrm{mul}}$.

- If $s \in \mathcal{V}$ then $s\sigma = t\sigma$. By taking $\mu = \{s \mapsto t\}$ we have $\mu \in \langle s \approx t \rangle$ and $\sigma = \mu\sigma$. Analogously, the case of $s \notin \mathcal{V}$ and $t \in \mathcal{V}$ is proved.
- If $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$ with $f \notin \mathcal{F}_{\mathsf{A}}$ then $s_i\sigma \leftrightarrow_{\mathsf{A}}^* t_i\sigma$. By the induction hypothesis for each $i \in \{1, \ldots, n\}$ there are $\mu_i \in \langle s_i \approx t_i \rangle$ and τ_i such that $\mu_i\tau_i \leftrightarrow_{\mathsf{A}}^* \sigma$. We define $\mu = \mu_1 \cdots \mu_n$ and

  $$\tau(x) = \begin{cases} x\tau_i & \text{if } 1 \leqslant i \leqslant n \text{ and } x \in \mathcal{V}\mathsf{ar}(s_i) \cup \mathcal{V}\mathsf{ar}(t_i) \\ x\sigma & \text{otherwise} \end{cases}$$

  It is not difficult to prove $\mu \in \langle s \approx t \rangle$ and $\mu\tau \leftrightarrow_{\mathsf{A}}^* \sigma$.
- Otherwise, $s = f(s_1, s_2)$, $t = f(t_1, t_2)$ with $f \in \mathcal{F}_{\mathsf{A}}$. We distinguish three cases, depending on s_1 and t_1. Note that the root symbols of s_1 and t_1 are not in $\mathcal{F}_{\mathsf{A}}$, due to the assumption $s, t \in \mathsf{NF}(\mathsf{A}')$.
  - If $s_i \leftrightarrow_{\mathsf{A}}^* t_i$ for each $i \in \{1, 2\}$ then the claim is straightforward from the induction hypotheses.
  - If $s_1 \in \mathcal{V}$ and $s_1\sigma \leftrightarrow_{\mathsf{A}}^* f(t_1\sigma, u)$ for some term u then

    $$s\tau_1\sigma = f(u, s_2\sigma) \leftrightarrow_{\mathsf{A}}^* f(u, t_2\sigma)$$

    for $\tau_1 = \{s_1 \mapsto u\}$. By the induction hypothesis $\mu_1 \in \langle s \approx t_2 \rangle$ and $\tau_1\sigma \leftrightarrow_{\mathsf{A}}^* \mu_1\tau_2$ for some substitutions μ_1 and τ_2. Let $\mu = \{s_1 \mapsto f(t_1, s_1)\}\mu_1$ and $\tau = \tau_1\tau_2$. We have $\mu \in \langle s \approx t \rangle$ and $\mu\tau \leftrightarrow_{\mathsf{A}}^* \{s_1 \mapsto f(t_1, u)\}\sigma \leftrightarrow_{\mathsf{A}}^* \sigma$.
  - If $t_1 \in \mathcal{V}$ and $t_1\sigma \leftrightarrow_{\mathsf{A}}^* f(s_1\sigma, u)$ for some u then the above argument goes through.

□

Theorem 3.6. *Let* s *and* t *be linear terms with* $\mathcal{V}\mathsf{ar}(s) \cap \mathcal{V}\mathsf{ar}(t) = \varnothing$. *The set* $\langle s{\downarrow_{\mathsf{A}'}} \approx t{\downarrow_{\mathsf{A}'}} \rangle$ *is a finite complete set of their* A*-unifiers.*

Proof. Finiteness is trivial. Completeness is shown by Lemma 3.4 and Lemma 3.5. □

We illustrate use of the theorem. Let $s = \mathsf{0} + x$, $t = (x' + y') + z'$, and $\mathcal{F}_{\mathsf{A}} = \{+\}$. A complete set of A-unifiers for the terms is computed as follows:

$$\begin{aligned} \langle s{\downarrow_{\mathsf{A}'}} \approx t{\downarrow_{\mathsf{A}'}} \rangle &= \langle \mathsf{0} + x \approx x' + (y' + z') \rangle \\ &= \big(\langle \mathsf{0} \approx x' \rangle \langle x \approx y' + z' \rangle\big) \cup \big(\{\{x' \mapsto \mathsf{0} + x'\}\} \langle x \approx x' + (y' + z') \rangle\big) \\ &= \big\{\{x' \mapsto \mathsf{0}, x \mapsto y' + z'\},\ \{x' \mapsto \mathsf{0} + x', x \mapsto x' + (y' + z')\}\big\} \end{aligned}$$

This set induces the A-critical pairs:

$$\begin{array}{rcl} y' + z' & {}_{\{1\},\mathsf{A}}\!\leftarrow \rtimes \rightarrow_{\{3\}} & \mathsf{0} + (y' + z') \\ x' + (y' + z') & {}_{\{1\},\mathsf{A}}\!\leftarrow \rtimes \rightarrow_{\{3\}} & (\mathsf{0} + x') + (y' + z') \end{array}$$

Both of the right-hand sides reduces to the corresponding left-hand sides by the rewriting modulo step $\to_{\{1\},\mathsf{A}}$.

3.2 Coherence

Consider the A-critical pair:

$$x + z \;\; {}_{\{1\},\mathsf{A}}\!\leftarrow \rtimes \rightarrow_{\{2\}} \;\; (x + \mathsf{0}) + z$$

Contrary to our intention, $(x + \mathsf{0}) + z \to_{\{1\},\mathsf{A}} x + z$ does not hold, and thus $\mathsf{CR}(\{1\}, \mathsf{A})$ is refuted by Theorem 3.1. The undesired incapability of rewriting is known as the coherence problem of rewriting modulo [10, 14].

Definition 3.7. *A pair* $(\mathcal{R}, \mathcal{E})$ *is* strongly coherent *if* $\leftrightarrow_{\mathcal{E}}^* \cdot \to_{\mathcal{R},\mathcal{E}} \subseteq \to_{\mathcal{R},\mathcal{E}} \cdot \leftrightarrow_{\mathcal{E}}^*$.

Lemma 3.8. *Suppose* $\mathcal{E}$ *is reversible and* $(\mathcal{R}, \mathcal{E})$ *is strongly coherent. If* $\mathsf{CR}(\mathcal{R}, \mathcal{E})$ *then* $\mathcal{R} \cup \mathcal{E}$ *is confluent, and vice versa.* □

While the strong coherence property always holds for rewriting modulo C, rewriting modulo A and AC rarely satisfy the property. This can be overcome by *extending* a rewrite system. While an extension for AC is well-known [10, 14], we present the A-extension of a TRS.

Definition 3.9. *Let* $\mathcal{R}$ *be a TRS. The* A-extended *TRS* $\mathsf{Ext}_{\mathsf{A}}(\mathcal{R})$ *consists of*

$$\ell \to r \qquad f(\ell, x) \to f(r, x) \qquad f(x, f(\ell, y)) \to f(x, f(r, y))$$
$$f(x, \ell) \to f(x, r)$$

for all rules $\ell \to r \in \mathcal{R}$ *with* $f = \mathrm{root}(\ell) \in \mathcal{F}_A$. *Here* x *and* y *are fresh variables not in* ℓ.

Lemma 3.10. *The pair* $(\mathsf{Ext}_{\mathsf{A}}(\mathcal{R}), \mathsf{A})$ *is strongly coherent and* $\to_{\mathsf{Ext}_{\mathsf{A}}(\mathcal{R})} = \to_{\mathcal{R}}$.

Proof. From the inclusion $\to_{\mathsf{A}} \cdot \to_{\mathsf{Ext}_{\mathsf{A}}(\mathcal{R}),\mathsf{A}} \subseteq \to_{\mathsf{Ext}_{\mathsf{A}}(\mathcal{R}),\mathsf{A}} \cdot \to_{\mathsf{A}}^*$ the first claim follows. Since $\to_{\mathcal{R}}$ is closed under contexts, the second claim is trivial. □

The TRS $\mathsf{Ext}_{\mathsf{A}}(\{1\})$ consists of the four rules:

$$\mathsf{0} + x \to x \qquad w + ((\mathsf{0} + x) \to w + x$$
$$(\mathsf{0} + x) + y \to x + y \qquad w + ((\mathsf{0} + x) + y) \to w + (x + y)$$

As the extended TRS contains all original rules, we have again the previous A-critical pair:

$$x + z \;\; {}_{\{1\},\mathsf{A}}\!\leftarrow \rtimes \rightarrow_{\{2\}} \;\; (x + \mathsf{0}) + z$$

Since $(x + \mathsf{0}) + z \to_{\mathsf{Ext}_{\mathsf{A}}(\{1\}),\mathsf{A}} x + z$ holds, the pair is joinable. Similarly, one can verify that all other A-critical pairs are joinable. Therefore, $\mathsf{CR}(\mathsf{Ext}_{\mathsf{A}}(\{1\}), \mathsf{A})$ is concluded by Theorem 3.1. Finally, confluence of $\mathcal{R}_1$ is established.

3.3 Commutative Unification

Commutative unification also benefits from left-linearity. We define $\mathcal{U}^{\mathsf{C}}_{\mathcal{E}}(s \approx t)$ as $\{\mu \mid s \mathrel{-\!\!\circ\!\!\to_{\mathsf{C}}} s' \text{ and } \mu \in \mathcal{U}_{\mathcal{E}}(s' \approx t) \text{ for some } s'\}$.

Lemma 3.11. *Suppose* C *and* $\mathcal{E} \cup \mathcal{E}^{-1}$ *commute. If* $\mathcal{V}\mathsf{ar}(s) \cap \mathcal{V}\mathsf{ar}(t) = \varnothing$ *and* s *is linear then* $\mathcal{U}^{\mathsf{C}}_{\mathcal{E}}(s \approx t)$ *is a complete set of* $\mathsf{C} \cup \mathcal{E}$*-unifiers for* s *and* t*.*

Proof. Since it is trivial that $\mathcal{U}^{\mathsf{C}}_{\mathcal{E}}(s \approx t)$ consists of $\mathcal{E} \cup \mathsf{C}$-unifiers, we only show completeness of the set. Let $s\sigma \leftrightarrow^*_{\mathsf{C}\cup\mathcal{E}} t\sigma$. By commutation we have $s\sigma \mathrel{{}^*_{\mathsf{C}}\!\!\leftarrow} u \leftrightarrow^*_{\mathcal{E}} t\sigma$ for some u. Since $\mathrel{-\!\!\circ\!\!\to_{\mathsf{C}}} = \leftrightarrow^*_{\mathsf{C}}$ holds, $s\sigma \mathrel{-\!\!\circ\!\!\to_{\mathsf{C}}} u$. Due to the linearity of s there are s' and σ' such that $u = s'\sigma'$, $s \mathrel{-\!\!\circ\!\!\to_{\mathsf{C}}} s'$, and $x\sigma \mathrel{-\!\!\circ\!\!\to_{\mathsf{C}}} x\sigma'$ for all variables x. We define μ as follows:

$$\mu(x) = \begin{cases} x\sigma & \text{if } x \in \mathcal{V}\mathsf{ar}(s) \\ x\sigma' & \text{if } x \in \mathcal{V}\mathsf{ar}(t) \\ x & \text{otherwise} \end{cases}$$

Since s and t share no variables, μ is well-defined. By the definition we obtain $s'\mu \leftrightarrow^*_{\mathcal{E}} t\mu$. □

Since A and C are left-linear TRSs that share no function symbols, their commutation can be proved (by using e.g. Theorem 4.1 in the next chapter). So Lemma 3.11 gives a way to compute $\mathsf{A} \cup \mathsf{C}$-critical pairs.

Example 3.12. *Consider the left-linear TRS* $\mathcal{R}_2$ *with* $\mathcal{F}_{\mathsf{A}} = \{*\}$ *and* $\mathcal{F}_{\mathsf{C}} = \{\mathsf{eq}\}$*:*

$$\begin{array}{llll} 1\colon & \mathsf{eq}(\mathsf{a},\mathsf{a}) \to \mathsf{T} & 3\colon & \mathsf{eq}(\mathsf{a} * x, y * \mathsf{a}) \to \mathsf{eq}(x,y) \\ 2\colon & \mathsf{eq}(\mathsf{a}, x * y) \to \mathsf{F} & 4\colon & \mathsf{eq}(x,y) \to \mathsf{eq}(y,x) \end{array} \qquad \begin{array}{ll} 5\colon & (x * y) * z \to x * (y * z) \\ 6\colon & x * (y * z) \to (x * y) * z \end{array}$$

Let $\mathcal{R} = \{1,2,3\}$ *and* $\mathcal{E} = \{4,5,6\}$*. Note that* $\mathcal{E} = \mathsf{C} \cup \mathsf{A}$*. It is sufficient to show* $\mathsf{CR}(\mathsf{Ext}_{\mathsf{A}}(\mathcal{R}), \mathcal{E})$*. We can use AC-RPO to prove termination of* $\mathcal{R}/\mathcal{E}$*, which is equivalent to that of* $\mathsf{Ext}_{\mathsf{A}}(\mathcal{R})/\mathcal{E}$ *due to the identity in Lemma 3.8. Let* $s = \mathsf{eq}(\mathsf{a} * x, y * \mathsf{a})$ *and* $t = \mathsf{eq}(\mathsf{a} * x', y' * \mathsf{a})$*. A complete set of their* $\mathsf{A} \cup \mathsf{C}$*-unifiers is:*

$$\begin{aligned} \mathcal{U}^{\mathsf{C}}_{\mathsf{A}}(s \approx t) &= \langle s \approx t \rangle \cup \langle \mathsf{eq}(y * \mathsf{a}, \mathsf{a} * x) \approx \mathsf{eq}(\mathsf{a} * x', y' * \mathsf{a}) \rangle \\ &= \left\{ \begin{array}{llll} \{x \mapsto x', & y \mapsto y'\}, & & \\ \{x \mapsto \mathsf{a}, & y \mapsto \mathsf{a}, & x' \mapsto \mathsf{a}, & y' \mapsto \mathsf{a}\} \\ \{x \mapsto y' * \mathsf{a}, & y \mapsto \mathsf{a} * y, & x' \mapsto y * \mathsf{a}, & y' \mapsto \mathsf{a} * y'\} \end{array} \right\} \end{aligned}$$

In this way we can compute complete sets to induce the set of all $\mathcal{E}$*-critical pairs. Since all pairs are joinable,* $\mathsf{CR}(\mathcal{R}, \mathcal{E})$ *is concluded.*

Chapter 4

Commutation

4.1 Commutation Criteria

Our tool employs three commutation criteria. The first commutation criterion is the *development closedness theorem* [3, 9, 20, 22].

Theorem 4.1. *Left-linear TRSs $\mathcal{R}$ and $\mathcal{S}$ commute if the inclusions*

$$ {}_{\mathcal{R}}{\leftarrow}{\rtimes}{\rightarrow}_{\mathcal{S}} \subseteq {\multimap}_{\mathcal{S}} \qquad\qquad {}_{\mathcal{R}}{\leftarrow}{\ltimes}{\rightarrow}_{\mathcal{S}} \subseteq {\rightarrow}^*_{\mathcal{S}} \cdot {\leftarrow\!\!\circ}_{\mathcal{R}} $$

hold. □

The second criterion is the commutation version of the rule labeling technique with weight function [23, 1].

Definition 4.2. *A* weight function *w is a function from $\mathcal{F}$ to $\mathbb{N}$. The weight $w(C)$ of a context C is defined as follows:*

$$ w(C) = \begin{cases} 0 & \text{if } C = \Box \\ w(f) + w(C') & \text{if } C = f(t_1, \ldots, C', \ldots, t_n) \text{ with a context } C' \end{cases} $$

The weight is admissible *for a TRS $\mathcal{R}$ if*

$$ \{w(C) \mid C[x] = \ell\} \geqslant^{\mathrm{mul}} \{w(C) \mid C[x] = r\} $$

holds for all $\ell \to r \in \mathcal{R}$ and $x \in \mathcal{V}\mathsf{ar}(r)$. A rule labeling *$\phi$ for a TRS $\mathcal{R}$ is a function from rules in $\mathcal{R}$ to $\mathbb{N}$. The labeled step $\xrightarrow{\alpha}_{\mathcal{R}}$ is defined as follows: $s \to_{\mathcal{R},(k,m)} t$ if there are a rule $\ell \to r$, a context C, and a substitution σ such that $s = C[\ell\sigma]$, $t = C[r\sigma]$, and $\alpha = (w(C), \phi(\ell \to r))$.*

In the next theorem we use the following abbreviations for labeled steps:

$$ \xrightarrow{I} = \bigcup_{\alpha \in I} \xrightarrow{\alpha} \qquad\qquad \Upsilon\alpha = \{\beta \in I \mid \alpha \succ \beta\} \qquad\qquad \Upsilon\alpha\beta = (\Upsilon\alpha) \cup (\Upsilon\beta) $$

where, $\succ$ stands for the lexicographic order on $\mathbb{N} \times \mathbb{N}$.

Theorem 4.3. *Left-linear TRSs $\mathcal{R}$ and $\mathcal{S}$ commute if there is an admissible weight function w and a rule labeling ϕ for $\mathcal{R} \cup \mathcal{S}$ such that*

$$({}_{\mathcal{R}}\overset{\alpha}{\leftarrow}\rtimes\overset{\beta}{\rightarrow}_{\mathcal{S}}) \cup ({}_{\mathcal{R}}\overset{\alpha}{\leftarrow}\ltimes\overset{\beta}{\rightarrow}_{\mathcal{S}}) \subseteq \overset{\Upsilon\alpha}{\longrightarrow}^{*}_{\mathcal{S}} \cdot \overset{\beta}{\rightarrow}^{=}_{\mathcal{S}} \cdot \overset{\Upsilon\alpha\beta}{\longrightarrow}^{*}_{\mathcal{S}} \cdot \overset{\Upsilon\alpha\beta}{\longleftarrow}^{*}_{\mathcal{R}} \cdot \overset{\alpha}{\leftarrow}^{=}_{\mathcal{R}} \cdot \overset{\Upsilon\beta}{\longleftarrow}^{*}_{\mathcal{R}}$$

for all pairs $\alpha, \beta \in \mathbb{N} \times \mathbb{N}$. □

The final criterion is a trivial adaptation of Theorem 3.1 to the commutation property, integrating lemmata in Section 3.

Theorem 4.4. *Let $\mathcal{R}, \mathcal{S}$ be left-linear TRSs and $\mathcal{E} \in \{\mathsf{A}, \mathsf{AC}\}$ such that $\mathcal{R}/(\mathcal{E} \cup \mathsf{C})$ is terminating. The TRSs $\mathcal{R} \cup \mathcal{E}'$ and $\mathcal{S} \cup \mathcal{E}'$ commute if and only if*

$$({}_{\mathcal{R}',\mathcal{E}'}\leftarrow\rtimes\rightarrow_{\mathcal{S}'\cup\mathcal{E}'}) \cup ({}_{\mathcal{R}'\cup\mathcal{E}'}\leftarrow\ltimes\rightarrow_{\mathcal{S}',\mathcal{E}'}) \subseteq \rightarrow^{*}_{\mathcal{S}',\mathcal{E}'} \cdot \leftrightarrow^{*}_{\mathcal{E}'} \cdot {}^{*}_{\mathcal{R}',\mathcal{E}'}\leftarrow$$

where, $\mathcal{R}' = \mathsf{Ext}_{\mathcal{E}}(\mathcal{R})$, $\mathcal{S}' = \mathsf{Ext}_{\mathcal{E}}(\mathcal{S})$, and $\mathcal{E}' = \mathcal{E} \cup \mathsf{C}$. □

Note that our tool uses the algorithm in [16] for AC unification and flattened term representation for overcoming the coherence problem of AC-rewriting. Since we use the dedicated algorithms for A and AC unification, currently we cannot employ Theorem 3.1 with $\mathcal{E} = \mathsf{A} \cup \mathsf{AC}$.

4.2 Commutation Theorem

The next theorem is known as Hindley's Commutation Theorem [7].

Theorem 4.5. *ARSs $\mathcal{A} = (A, \{\rightarrow_\alpha\}_{\alpha\in I})$ and $\mathcal{B} = (B, \{\rightarrow_\beta\}_{\beta\in J})$ commute if $\rightarrow_\alpha$ and $\rightarrow_\beta$ commute for all $\alpha \in I$ and $\beta \in J$.* □

Example 4.6. *Consider the left-linear TRS $\mathcal{R}_3$:*

1:	$\mathsf{0} \times y \to \mathsf{nil}$	9:	$\mathsf{q}(\mathsf{q}(x,y),z) \to \mathsf{q}(x,\mathsf{q}(y,z))$
2:	$\mathsf{s}(x) \times y \to y +\!\!+ (x \times y)$	10:	$\mathsf{q}(x,\mathsf{q}(y,z)) \to \mathsf{q}(\mathsf{q}(x,y),z)$
3:	$\mathsf{hd}(\mathsf{c}(x)) \to x$	11:	$\mathsf{q}(e,x) \to x$
4:	$\mathsf{hd}(\mathsf{c}(x) +\!\!+ y) \to x$	12:	$\mathsf{q}(x,\mathsf{e}) \to x$
5:	$\mathsf{nil} +\!\!+ x \to x$	13:	$x + (y + z) \to (x + y + z)$
6:	$x +\!\!+ \mathsf{nil} \to x$	14:	$x + y \to y + x$
7:	$x +\!\!+ (y +\!\!+ z) \to (x +\!\!+ y) +\!\!+ z$	15:	$\mathsf{0} + x \to x$
8:	$(x +\!\!+ y) +\!\!+ z \to x +\!\!+ (y +\!\!+ z)$	16:	$\mathsf{s}(x + y) \to s(x + y)$
		17:	$\mathsf{len}(\mathsf{e}) \to \mathsf{0}$
		18:	$\mathsf{len}(\mathsf{a}) \to s(\mathsf{0})$
		19:	$\mathsf{len}(\mathsf{q}(\mathsf{a}, y)) \to \mathsf{len}(\mathsf{a}) + \mathsf{len}(y)$

By using the Commutation Theorem we show self-commutation of $\mathcal{R}_3$. Suppose $\mathcal{R}_{3A} = \{1, \ldots, 8\}$ and $\mathcal{R}_{3B} = \{9, \ldots, 19\}$.

1. *Self-commutation of* $\mathcal{R}_{3A}$ *follows from* $\mathsf{CR}(\mathsf{Ext}_{\{7,8\}}(\{1,\ldots,6\}),\{7,8\})$.
2. *We show self-commutation of* $\mathcal{R}_{3B}$*:*
   (i) *Commutation of* $\mathcal{R}_{3B}$ *and* $\{10\}$ *follows from the rule labeling technique, and also* $\mathcal{R}_{3B}\setminus\{10\}$ *and* $\{10\}$ *commute by the same way.*
   (ii) *Self-commutation of* $\mathcal{R}_{3B}\setminus\{10\}$ *is proved by the next Church-Rosser modulo property:* $\mathsf{CR}(\mathsf{Ext}_{\{13,14\}}\{9,11,12,15,\ldots,19\},\{13,14\})$.
   (iii) *By the Commutation Theorem commutation of* $\mathcal{R}_{3B}$ *and* $\mathcal{R}_{3B}\setminus\{10\}$ *is proved, and hence* $\mathcal{R}_{3B}$ *and* $\mathcal{R}_{3B}$ *commute.*
3. *Commutation of* $\mathcal{R}_{3A}$ *and* $\mathcal{R}_{3B}$ *is proved by the development closedness.*

Hence, $\mathcal{R}_3$ *is confluent.*

It is a non-trivial task to find suitable commuting subsystems from an exponential number of candidates. In order to address the problem we introduce a decomposition method based on *composability*, which was introduced by Ohlebusch [13]. Let $\mathcal{R}$ be a TRS. We write $\mathcal{F}_\mathcal{R}$, $\mathcal{D}_\mathcal{R}$, and $\mathcal{C}_\mathcal{R}$ for the following sets:

$$\mathcal{F}_\mathcal{R}=\bigcup_{\ell\to r\in\mathcal{R}}\mathcal{F}\mathsf{un}(\ell)\cup\mathcal{F}\mathsf{un}(r)\qquad\mathcal{D}_\mathcal{R}=\{\mathrm{root}(\ell)\mid\ell\to r\in\mathcal{R}\}\qquad\mathcal{C}_\mathcal{R}=\mathcal{F}_\mathcal{R}\setminus\mathcal{D}_\mathcal{R}$$

Definition 4.7. *We say that TRSs* $\mathcal{R}$ *and* $\mathcal{S}$ *are* composable *if* $\mathcal{C}_\mathcal{R}\cap\mathcal{D}_\mathcal{S}=\mathcal{C}_\mathcal{S}\cap\mathcal{D}_\mathcal{R}=\varnothing$ *and* $\{\ell\to r\in\mathcal{R}\cup\mathcal{S}\mid\mathrm{root}(\ell)\in\mathcal{D}_\mathcal{R}\cup\mathcal{D}_\mathcal{S}\}\subseteq\mathcal{R}\cap\mathcal{S}$.

Ohlebusch [13] posed the following question.

Conjecture 4.8. *Left-linear composable TRSs* $\mathcal{R}$ *and* $\mathcal{S}$ *are confluent if and only if* $\mathcal{R}\cup\mathcal{S}$ *is confluent.*

Although this conjecture still remains open, the variation Theorem 4.11 is valid.

Lemma 4.9. *If TRSs* $\mathcal{R}$ *and* $\mathcal{S}$ *are composable then* $\mathcal{T}(\mathcal{F}_\mathcal{S},\mathcal{V})\subseteq\mathsf{NF}(\mathcal{R}\setminus\mathcal{S})$.

Proof. We claim $\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap\mathcal{D}_\mathcal{S}=\varnothing$. Because the composability implies $\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap\mathcal{C}_\mathcal{S}=\varnothing$, we obtain the following equalities from the claim:

$$\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap\mathcal{F}_\mathcal{S}=\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap(\mathcal{D}_\mathcal{S}\cup\mathcal{C}_\mathcal{S})=(\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap\mathcal{D}_\mathcal{S})\cup(\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap\mathcal{C}_\mathcal{S})=\varnothing$$

Therefore, $\mathcal{T}(\mathcal{F}_\mathcal{S},\mathcal{V})\subseteq\mathsf{NF}(\mathcal{R}\setminus\mathcal{S})$ is concluded. It remains to show the claim. Assume to the contrary there is an $f\in\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap\mathcal{D}_\mathcal{S}$. By the definition there is a rule $\ell\to r\in\mathcal{R}\setminus\mathcal{S}$ with $\mathrm{root}(\ell)=f$. Thus, the composability of $\mathcal{R}$ and $\mathcal{S}$ deduces $\ell\to r\in\mathcal{R}\cap\mathcal{S}$, which contracts to the assumption $\ell\to r\notin\mathcal{S}$. Hence, we obtain $\mathcal{D}_{\mathcal{R}\setminus\mathcal{S}}\cap\mathcal{D}_\mathcal{S}=\varnothing$. □

Lemma 4.10. *Composable TRSs* $\mathcal{R}$ *and* $\mathcal{S}$ *are confluent if* $\mathcal{R}\cup\mathcal{S}$ *is confluent.*

Proof. We only argue confluence of $\mathcal{R}$. Let $t, u \in \mathcal{T}(\mathcal{F}_{\mathcal{R}}, \mathcal{V})$ with $t \,{}^*_{\mathcal{R}}\!\leftarrow \cdot \rightarrow^*_{\mathcal{R}} u$. Because $\mathcal{R} \cup \mathcal{S}$ is confluent, we have $t \rightarrow^*_{\mathcal{R}\cup\mathcal{S}} \cdot \,{}^*_{\mathcal{R}\cup\mathcal{S}}\!\leftarrow u$. According to Lemma 4.9, every term over $\mathcal{F}_{\mathcal{S}}$ is irreducible by $\mathcal{S} \setminus \mathcal{R}$. Therefore, the valley must be of the form $t \rightarrow^*_{\mathcal{R}} \cdot \,{}^*_{\mathcal{R}}\!\leftarrow u$. □

Theorem 4.11. *Commuting composable TRSs $\mathcal{R}$ and $\mathcal{S}$ are confluent if and only if $\mathcal{R} \cup \mathcal{S}$ is confluent.*

Proof. The proof is obtained by combining the Commutation Theorem and Lemma 4.10. □

Example 4.12. *Recall the TRS $\mathcal{R}_3$ from Example 4.6. It is the union of the four commuting composable subsystems: $\{1, 2, 5, 6, 7, 8\}$, $\{3, 4, \ldots, 8\}$, $\{9\}$, and $\mathcal{R}_{3B}$. Confluence of each subsystem can be proved in the same method used in the previous example. Hence, $\mathcal{R}_3$ is confluent.*

Chapter 5

Implementation

All presented techniques have been implemented in CoLL version 1.1. The tool consists of about 5,000 lines of OCaml code. Given an input TRS, the tool first performs the next trivial *redundant rule elimination*.

Theorem 5.1. *Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs with $\mathcal{S} \subseteq \to_{\mathcal{R}}^*$. The TRS $\mathcal{R} \cup \mathcal{S}$ is confluent if and only if $\mathcal{R}$ is confluent.* □

Example 5.2. *We illustrate the elimination technique with the small example taken from the Confluence Problem Database (Cops)[1]: Consider the TRS:*

$$1\colon \mathsf{f}(x) \to \mathsf{g}(x, \mathsf{f}(x)) \qquad 2\colon \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(x)))) \to \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{g}(x, \mathsf{f}(x)))))$$

Since $\{2\} \subseteq \to_{\{1\}}^*$ holds, we eliminate the redundant rule 2. Confluence of the simplified system $\{1\}$ is easily shown by Theorem 4.1. Note that CoLL cannot prove confluence without using the elimination technique.

Next, the tool employs Theorem 4.11 to split the simplified TRS into commuting composable subsystems $\mathcal{R}_1, \ldots, \mathcal{R}_n$. For each subsystem $\mathcal{R}_i$ the tool checks non-confluence by the technique by Zankl et al. [24, Lemma 1]. If non-confluence is detected, the tool outputs NO (non-confluence is proved). Otherwise, the tool uses the Commutation Theorem together with the three commutation criteria (Theorems 4.1, 4.3, and 4.4) to determine self-commutation of $\mathcal{R}_i$. Suitable commuting subsystems are searched by enumeration. It outputs YES (confluence is proved) if all of $\mathcal{R}_1, \ldots, \mathcal{R}_n$ are confluent. Concerning automation, we employed AC-RPO for checking termination $\mathcal{R}/\mathcal{E}$ automatically. Automation of Theorem 4.3 is based on the SAT encoding technique of [8].

We tested the presented techniques on 188 left-linear TRSs in Cops Nos. 1–425, where we ruled out duplicated problems. We compared CoLL (v1.1) with the tools that participated in the 3rd Confluence Competition (CoCo 2014): ACP v0.5[2], CSI v0.4.1[3], and

[1] `http://cops.uibk.ac.at/`

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

[3] `http://cl-informatik.uibk.ac.at/software/csi/`

	YES	NO	timeout
Church-Rosser modulo	**93**	**8**	1
development closed	**17**	**0**	0
rule labeling	**58**	**0**	26
all three	**125**	**8**	–
all with elimination	**136**	**9**	–
CoLL	**136**	**16**	20
ACP	**134**	**41**	0
CSI	**118**	**38**	11
Saigawa	**105**	**16**	17

$\mathcal{E}$	YES	NO	timeout
$\varnothing$	**18**	**8**	0
A	**24**	**0**	0
C	**42**	**8**	0
AC	**64**	**8**	1
C $\uplus$ AC	**88**	**8**	1
A $\uplus$ C $\uplus$ AC	**93**	**8**	1

Figure 5.1: Experimental results.

Saigawa v1.7[4]. The tests were single-threaded run on a PC equipped with an Intel Core i7-4500U CPU with 1.8 GHz and 3.8 GB of RAM using a timeout of 120 seconds. We employed AC-RPO to check the termination condition for Theorem 4.4. The first table in Figure 5.1 summarizes results. The first three indicates the results of each commutation criterion without using the Commutation Theorem. The row 'all three' is the summation of their results, and 'all with elimination' is the same but the elimination technique is enabled. The second table indicates the results of individual theories for Theorem 4.4. The row CoLL corresponds to the strategy stated above. From the results it is unclear whether commutation is useful. This is because out of 188 TRSs, 136 are proved confluent by 'all with elimination' and 41 are known to be non-confluent. Therefore, stemming from functional programs, we prepared additional confluence problems to evaluate commutation. Its problems and results are available in Chapter A (TRSs $\mathcal{S}_1$ – $\mathcal{S}_{12}$) of Appendix.

[4] `http://www.jaist.ac.jp/project/saigawa/`

Chapter 6

Conclusion

We presented the new confluence tool CoLL for left-linear TRSs, which proves confluence via commutation. Our primary contribution is automation of Jouannaud and Kirchner's Church-Rosser modulo criterion for associativity and/or commutativity theory, where left-linearity and commutation are exploited in several ways.

CoLL is neither the first tool that implements the Church-Rosser modulo criterion nor decomposition techniques. The Church-Rosser checker CRC 3[1] [5] for Maude is capable of handling various AC-related theories except the solo use of associativity theory. It is worth investigating whether our unification algorithm can be integrated. Aoto and Toyama's *reduction-preserving completion* [2] extends a TRS by *adding* redundant rules induced from critical pairs. ACP proved that the method effectively works for TRSs including reversible rules. A remarkable advantage is that it only uses syntactical unification, so it is capable of handling non-AC reversible rules. A disadvantage of the method is that it hardly works for associativity theory. Layer-preserving decomposition [13] has been implemented in ACP and CSI. The technique is incomparable to Theorem 4.11. If it is affirmatively solved, Conjecture 4.8 generalizes those two results for the class of left-linear TRSs.

As future work we plan to investigate whether Theorem 4.11 can be generalized to cover a subclass of *hierarchical combination* [13]. Another interesting direction is the modularity of the commutation property. Since confluence is a modular property [19], it is closed under *signature extension*. Contrary to our expectation, (even local) commutation is *not* signature extensible. Consider the TRSs $\mathcal{R}$ and $\mathcal{S}$ over the signature $\mathcal{F} = \{\mathsf{f}^{(1)}, \mathsf{a}^{(0)}, \mathsf{b}^{(0)}\}$:

$$\mathcal{R} = \{\ \mathsf{a} \to \mathsf{b}\ \} \qquad \mathcal{S} = \{\ \mathsf{f}(x,x) \to \mathsf{b},\ \mathsf{f}(\mathsf{b},x) \to \mathsf{b},\ \mathsf{f}(x,\mathsf{b}) \to \mathsf{b}\ \}$$

Since $C[\mathsf{b}] \to_{\mathcal{S}}^{*} \mathsf{b}$ holds, we obtain the strong commutation ${}_{\mathcal{R}}\!\leftarrow \cdot \to_{\mathcal{S}} \subseteq \to_{\mathcal{S}}^{*} \cdot \overset{=}{}_{\mathcal{R}}\!\leftarrow$, which entails commutation of $\mathcal{R}$ and $\mathcal{S}$. However, if one extends the signature to $\mathcal{F} \cup \{\mathsf{g}^{(1)}\}$, no longer the local peak $\mathsf{f}(\mathsf{g}(\mathsf{b}), \mathsf{g}(\mathsf{a})) \ {}_{\mathcal{R}}\!\leftarrow \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \to_{\mathcal{S}} \mathsf{b}$ commutes. We conjecture that (local) commutation is closed under signature extension for left-linear TRSs.

[1] `http://maude.lcc.uma.es/CRChC/`

Acknowledgements. I am most grateful to my supervisor Prof. Nao Hirokawa for his guidance and valuable advice. I would like to thank Prof. Takahito Aoto and Prof. Yuki Chiba for their comments on Hindley's commutation. I also thank Prof. Kazuhiro Ogata for suggesting me to investigate the coherence property. Special thanks go to members in Hirokawa, Ogawa, and Terauchi laboratories for their valuable advices and supports.

Bibliography

[1] T. Aoto. Automated confluence proof by decreasing diagrams based on rule-labelling. In *Proc. 21st RTA*, volume 6 of *LNCS*, pages 7–16, 2010.

[2] T. Aoto and Y. Toyama. A reduction-preserving completion for proving confluence of non-terminating term rewriting systems. *LMCS*, 8(1):1–29, 2012.

[3] T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In *RTA 2009*, volume 5595 of *LNCS*, pages 93–102, 2009.

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

[5] Francisco Durán and José Meseguer. A church-rosser checker tool for conditional order-sorted equational maude specifications. In *Rewriting Logic and Its Applications*, pages 69–85. 2010.

[6] B. Felgenhauer. Deciding confluence of ground term rewrite systems in cubic time. In *Proc. 23rd RTA*, LIPIcs, pages 165–175, 2012.

[7] J. R. Hindley. *The Church-Rosser Property and a Result in Combinatory Logic*. PhD thesis, University of Newcastle-upon-Tyne, 1964.

[8] N. Hirokawa and A. Middeldorp. Decreasing diagrams and relative termination. *Journal of Automated Reasoning*, 47(4):481–501, 2011.

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

[10] J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4):1155–1194, 1986.

[11] D. Klein and N. Hirokawa. Confluence of non-left-linear TRSs via relative termination. In *Proc. 18th LPAR*, volume 7180 of *LNCS*, pages 258–273, 2012.

[12] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[13] E. Ohlebusch. *Modular Properties of Composable Term Rewriting Systems*. PhD thesis, Universität Bielefeld, 1994.

[14] G.E. Peterson and M.E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28(2):233–264, 1981.

[15] G. Plotkin. Building in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[16] L. Pottier. Minimal solutions of linear diophantine systems: Bounds and algorithms. In *Proc. 4th RTA*, volume 488 of *LNCS*, pages 162–173, 1991.

[17] A. Rubio. A fully syntactic AC-RPO. *Information and Computation*, 178(2):515–533, 2002.

[18] K.U. Schulz. Word unification and transformation of generalized equations. In *Word Equations and Related Topics*, volume 677 of *LNCS*, pages 150–176, 1993.

[19] Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, 1987.

[20] Y. Toyama. Commutativity of term rewriting systems. In K. Fuchi and L. Kott, editors, *Programming of Future Generation Computers II*, pages 393–407. North-Holland, 1988.

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

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

[23] V. van Oostrom. Confluence by decreasing diagrams converted. In A. Voronkov, editor, *Proc. 19th RTA*, volume 5117 of *LNCS*, pages 306–320, 2008.

[24] H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI — a confluence tool. In *Proc. 23rd CADE*, pages 499–505. Springer, 2011.

Appendix A

Additional Experiments

As in Chapter 5, we compared the confluence tools for the TRSs $\mathcal{S}_1$ – $\mathcal{S}_{12}$, listed below. Table A.1 shows the results of tools. A cell is ✓ if confluence of the column's TRS is proved/disproved by the row's tool. Otherwise, × is written.

Table A.1: Experimental results for commutation

	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$	$\mathcal{S}_{11}$	$\mathcal{S}_{12}$	sum
all with elimination	✓	✓	×	×	×	×	×	×	✓	✓	×	×	4
CoLL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	12
ACP	×	×	×	×	×	×	×	×	✓	×	×	×	1
CSI	×	×	×	×	×	×	×	×	✓	×	×	×	1
Saigawa	×	×	×	×	×	×	×	×	×	×	×	×	0

$\mathcal{S}_1$: The TRS $\mathcal{R}_1$ in Chapter 3.

$$1\colon 0 + x \to x \qquad 2\colon x + (y + z) \to (x + y) + z \qquad 3\colon (x + y) + z \to x + (y + z)$$

$\mathcal{S}_2$: The TRS $\mathcal{R}_2$ in Example 3.12.

$$\begin{array}{lll} 1\colon \ \mathsf{eq}(\mathsf{a},\mathsf{a}) \to \mathsf{T} & 3\colon \ \mathsf{eq}(\mathsf{a} * x, \mathsf{a} * y) \to \mathsf{eq}(x, y) & 5\colon \ (x * y) * z \to x * (y * z) \\ 2\colon \ \mathsf{eq}(\mathsf{a}, x * y) \to \mathsf{F} & 4\colon \ \mathsf{eq}(x, y) \to \mathsf{eq}(y, x) & 6\colon \ x * (y * z) \to (x * y) * z \end{array}$$

$\mathcal{S}_3$: A TRS like a functional program.

$$\begin{array}{ll} 1\colon \ 0 \times y \to \mathsf{nil} & 5\colon \ \mathsf{nil} +\!\!+ x \to x \\ 2\colon \ \mathsf{s}(x) \times y \to y +\!\!+ (x \times y) & 6\colon \ x +\!\!+ \mathsf{nil} \to x \\ 3\colon \ \mathsf{hd}(\mathsf{c}(x)) \to x & 7\colon \ x +\!\!+ (y +\!\!+ z) \to (x +\!\!+ y) +\!\!+ z \\ 4\colon \ \mathsf{hd}(\mathsf{c}(x) +\!\!+ y) \to x & 8\colon \ (x +\!\!+ y) +\!\!+ z \to x +\!\!+ (y +\!\!+ z) \\ 9\colon \ \mathsf{from}(x) \to x : \mathsf{from}(\mathsf{s}(x)) & \end{array}$$

$\mathcal{S}_4$: An non-confluent TRS.

$$
\begin{array}{rcrc}
1: & x + y & \to & y + x \\
2: & (x + y) + z & \to & x + (y + z) \\
3: & \mathsf{0} + y & \to & y \\
4: & \mathsf{s}(x) + y & \to & \mathsf{s}(x + y) \\
5: & (x \cdot y) \cdot z & \to & x \cdot (y \cdot z) \\
6: & x \cdot (y \cdot z) & \to & (x \cdot y) \cdot z \\
7: & \mathsf{a} \cdot (\mathsf{b} \cdot \mathsf{a}) & \to & (\mathsf{b} \cdot \mathsf{a}) \cdot \mathsf{b} \\
8: & x \times y & \to & y \times x \\
9: & \mathsf{0} \times y & \to & \mathsf{0} \\
10: & \mathsf{s}(x) \times y & \to & y + (x \times y) \\
11: & \mathsf{fib}(x) & \to & x +\!\!+ (\mathsf{fib}(x) + \mathsf{fib}(\mathsf{s}(x)))
\end{array}
$$

$\mathcal{S}_5$: A TRS whose subsystems are constructor sharing

$$
\begin{array}{rcrc}
1: & \mathsf{a} & \to & \mathsf{b} \\
2: & \mathsf{b} & \to & \mathsf{c} \\
3: & \mathsf{c} & \to & \mathsf{a} \\
4: & \mathsf{f}(\mathsf{s}(\mathsf{a})) & \to & \mathsf{a} \\
5: & \mathsf{f}(\mathsf{s}(\mathsf{b})) & \to & \mathsf{f}(\mathsf{s}(\mathsf{a})) \\
6: & \mathsf{f}(\mathsf{s}(\mathsf{c})) & \to & \mathsf{f}(\mathsf{s}(\mathsf{b})) \\
7: & x + y & \to & y + x \\
8: & (x + y) + z & \to & x + (y + z) \\
9: & \mathsf{0} + y & \to & x \\
10: & \mathsf{s}(x) + y & \to & \mathsf{s}(x + y)
\end{array}
$$

$\mathcal{S}_6$: A proof score in CafeOBJ.

$$
\begin{array}{rcrc}
1: & (x * y) * z & \to & x * (y * z) \\
2: & x * (y * z) & \to & (x * y) * z \\
3: & \mathsf{eq}(x, y) & \to & \mathsf{eq}(y, x) \\
4: & \mathsf{eq}(\mathsf{a}, \mathsf{a}) & \to & \mathsf{T} \\
5: & \mathsf{eq}(\mathsf{a}, x * y) & \to & \mathsf{F} \\
6: & \mathsf{eq}(\mathsf{a} * x, \mathsf{a} * y) & \to & \mathsf{eq}(x, y) \\
7: & \mathsf{lem1}(x) & \to & \mathsf{eq}(\mathsf{hd}(x) * \mathsf{tl}(x), x) \\
8: & \mathsf{lem2}(x * y) & \to & \mathsf{eq}(\mathsf{lt}(x * y), \mathsf{lt}(\mathsf{tl}(x * y))) \\
9: & \mathsf{rep}(\mathsf{0}, x) & \to & \mathsf{nil} \\
10: & \mathsf{rep}(\mathsf{s}(x), y) & \to & x * \mathsf{rep}(x, y) \\
11: & \mathsf{hd}(\mathsf{c}(x) * y) & \to & x \\
12: & \mathsf{lt}(x * \mathsf{c}(y)) & \to & y \\
13: & \mathsf{tl}(\mathsf{c}(x) * y) & \to & y \\
14: & x + y & \to & y + x \\
15: & (x + y) + z & \to & x + (y + z) \\
16: & \mathsf{0} + y & \to & x \\
17: & \mathsf{s}(x) + y & \to & \mathsf{s}(x + y)
\end{array}
$$

$\mathcal{S}_7$: A Composable TRS.

$$
\begin{array}{rcrc}
1: & x +\!\!+ (y +\!\!+ z) & \to & (x +\!\!+ y) +\!\!+ z \\
2: & (x +\!\!+ y) +\!\!+ z & \to & x +\!\!+ (y +\!\!+ z) \\
3: & \mathsf{0} +\!\!+ x & \to & x \\
4: & x + y & \to & y + x \\
5: & (x + y) + z & \to & x + (y + z) \\
7: & \mathsf{s}(x) + y & \to & \mathsf{s}(x + y) \\
6: & \mathsf{0} + y & \to & y
\end{array}
$$

$\mathcal{S}_8$: **A TRS that admits no composable subsystems.**

$$
\begin{array}{rrrr}
1\colon & \mathsf{e}(x) \to x \mathbin{+\!\!+} \mathsf{e}(x+x) & 5\colon & x+y \to y+x \\
2\colon & x \mathbin{+\!\!+} (y \mathbin{+\!\!+} z) \to (x \mathbin{+\!\!+} y) \mathbin{+\!\!+} z & 6\colon & (x+y)+z \to x+(y+z) \\
3\colon & (x \mathbin{+\!\!+} y) \mathbin{+\!\!+} z \to x \mathbin{+\!\!+} (y \mathbin{+\!\!+} z) & 8\colon & \mathsf{s}(x)+y \to \mathsf{s}(x+y) \\
4\colon & \mathsf{0} \mathbin{+\!\!+} x \to x & 7\colon & \mathsf{0}+y \to y
\end{array}
$$

$\mathcal{S}_9$: **Queue, not confluent.**

$$
\begin{array}{rrrr}
1\colon & \mathsf{q}(\mathsf{e},x) \to x & 5\colon & \mathsf{eq}(\mathsf{e},\mathsf{q}(\mathsf{a},x)) \to \mathsf{false} \\
2\colon & \mathsf{q}(x,\mathsf{e}) \to x & 6\colon & \mathsf{eq}(\mathsf{q}(\mathsf{a},x),\mathsf{q}(\mathsf{a},y)) \to \mathsf{eq}(x,y) \\
3\colon & \mathsf{q}(\mathsf{q}(x,y),z) \to \mathsf{q}(x,\mathsf{q}(y,z)) & 7\colon & \mathsf{eq}(x,y) \to \mathsf{eq}(y,x) \\
4\colon & \mathsf{q}(x,\mathsf{q}(y,z)) \to \mathsf{q}(\mathsf{q}(x,y),z) &&
\end{array}
$$

$\mathcal{S}_{10}$: **Queue.**

$$
\begin{array}{rrrr}
1\colon & \mathsf{q}(\mathsf{e},x) \to x & 4\colon & \mathsf{eq}(\mathsf{e},\mathsf{q}(\mathsf{a},x)) \to \mathsf{false} \\
2\colon & \mathsf{q}(\mathsf{q}(x,y),z) \to \mathsf{q}(x,\mathsf{q}(y,z)) & 5\colon & \mathsf{eq}(\mathsf{q}(\mathsf{a},x),\mathsf{q}(\mathsf{a},y)) \to \mathsf{eq}(x,y) \\
3\colon & \mathsf{q}(x,\mathsf{q}(y,z)) \to \mathsf{q}(\mathsf{q}(x,y),z) & 6\colon & \mathsf{eq}(x,y) \to \mathsf{eq}(y,x)
\end{array}
$$

$\mathcal{S}_{11}$: **The length function for queues.**

$$
\begin{array}{rrrr}
1\colon & \mathsf{q}(\mathsf{q}(x,y),z) \to \mathsf{q}(x,\mathsf{q}(y,z)) & 7\colon & \mathsf{0}+x \to x \\
2\colon & \mathsf{q}(x,\mathsf{q}(y,z)) \to \mathsf{q}(\mathsf{q}(x,y),z) & 8\colon & \mathsf{s}(x)+y \to \mathsf{s}(x+y) \\
3\colon & \mathsf{q}(\mathsf{e},x) \to x & 9\colon & \mathsf{len}(\mathsf{e}) \to \mathsf{0} \\
4\colon & \mathsf{q}(x,\mathsf{e}) \to x & 10\colon & \mathsf{len}(\mathsf{a}) \to \mathsf{s}(\mathsf{0}) \\
5\colon & x+(y+z) \to (x+y)+z & 11\colon & \mathsf{len}(\mathsf{q}(\mathsf{a},y)) \to \mathsf{len}(\mathsf{a})+\mathsf{len}(y) \\
6\colon & x+y \to y+x &&
\end{array}
$$

$\mathcal{S}_{12}$: **The TRS $\mathcal{R}_3$ in Example 4.6.**

$$
\begin{array}{rrrr}
1\colon & \mathsf{0} \times y \to \mathsf{nil} & 12\colon & \mathsf{q}(\mathsf{q}(x,y),z) \to \mathsf{q}(x,\mathsf{q}(y,z)) \\
2\colon & \mathsf{s}(x) \times y \to y \mathbin{+\!\!+} (x \times y) & 13\colon & \mathsf{q}(x,\mathsf{q}(y,z)) \to \mathsf{q}(\mathsf{q}(x,y),z) \\
3\colon & \mathsf{hd}(\mathsf{c}(x)) \to x & 14\colon & \mathsf{q}(\mathsf{e},x) \to x \\
4\colon & \mathsf{hd}(\mathsf{c}(x) \mathbin{+\!\!+} y) \to x & 15\colon & \mathsf{q}(x,\mathsf{e}) \to x \\
5\colon & \mathsf{nil} \mathbin{+\!\!+} x \to x & 16\colon & x+(y+z) \to (x+y)+z \\
6\colon & x \mathbin{+\!\!+} \mathsf{nil} \to x & 17\colon & x+y \to y+x \\
7\colon & x \mathbin{+\!\!+} (y \mathbin{+\!\!+} z) \to (x \mathbin{+\!\!+} y) \mathbin{+\!\!+} z & 18\colon & \mathsf{0}+x \to x \\
8\colon & (x \mathbin{+\!\!+} y) \mathbin{+\!\!+} z \to x \mathbin{+\!\!+} (y \mathbin{+\!\!+} z) & 19\colon & \mathsf{s}(x)+y \to \mathsf{s}(x+y) \\
&& 20\colon & \mathsf{len}(\mathsf{e}) \to \mathsf{0} \\
&& 21\colon & \mathsf{len}(\mathsf{a}) \to \mathsf{s}(\mathsf{0}) \\
&& 22\colon & \mathsf{len}(\mathsf{q}(\mathsf{a},y)) \to \mathsf{len}(\mathsf{a})+\mathsf{len}(y)
\end{array}
$$

Appendix B

Examples

In this appendix we show examples of input and output for the confluence tool CoLL.

Listing B.1: Sample Input I.

```
(VAR x y z)
(RULES
  *(x,*(y,z))        -> *(*(x,y),z)
  *(*(x,y),z)        -> *(x,*(y,z))
  eq(*(a,x),*(a,y)) -> eq(x,y)
  eq(a,a)            -> T
  eq(a,*(x,y))       -> F
  eq(x,y)            -> eq(y,x)
)
```

Listing B.2: Sample Output I.

```
YES

1 decompositions
 #1 -----------
   1: *(x1,*(x2,x3)) -> *(*(x1,x2),x3)
   2: *(*(x1,x2),x3) -> *(x1,*(x2,x3))
   3: eq(*(a(),x1),*(a(),x2)) -> eq(x1,x2)
   4: eq(a(),a()) -> T()
   5: eq(a(),*(x1,x2)) -> F()
   6: eq(x1,x2) -> eq(x2,x1)

@Jouannaud and Kirchner's criterion
--- R
   1: *(x1,*(x2,x3)) -> *(*(x1,x2),x3)
   2: *(*(x1,x2),x3) -> *(x1,*(x2,x3))
   3: eq(*(a(),x1),*(a(),x2)) -> eq(x1,x2)
   4: eq(a(),a()) -> T()
   5: eq(a(),*(x1,x2)) -> F()
   6: eq(x1,x2) -> eq(x2,x1)

--- S
   1: *(x1,*(x2,x3)) -> *(*(x1,x2),x3)
   2: *(*(x1,x2),x3) -> *(x1,*(x2,x3))
   3: eq(*(a(),x1),*(a(),x2)) -> eq(x1,x2)
   4: eq(a(),a()) -> T()
   5: eq(a(),*(x1,x2)) -> F()
   6: eq(x1,x2) -> eq(x2,x1)
```

Listing B.3: Sample Input II.

```
(VAR x y z)
(RULES
  ++(++(x,y),z) -> ++(x,++(y,z))
  ++(x,++(y,z)) -> ++(++(x,y),z)
  ++(nil,x) -> x
  ++(x,nil) -> x

  hd(c(x))   -> x
  hd(++(c(x),y)) -> x

  *(nil,y)    -> nil
  *(s(x),y) -> ++(y,*(x,y))

  q(q(x,y),z) -> q(x,q(y,z))
  q(x,q(y,z)) -> q(q(x,y),z)
  q(e,x) -> x
  q(x,e) -> x

  plus(x,plus(y,z)) -> plus(plus(x,y),z)
  plus(x,y) -> plus(y,x)
  plus(0,x) -> x
  plus(s(x),y) -> s(plus(x,y))

  len(e) -> 0
  len(a) -> s(0)
  len(q(a,y)) -> plus(len(a),len(y))
)
```

Listing B.4: Sample Output II.

```
YES

3 decompositions
 #1 -----------
   1: ++(++(x1,x2),x3) -> ++(x1,++(x2,x3))
   2: ++(x1,++(x2,x3)) -> ++(++(x1,x2),x3)
   3: ++(nil(),x1) -> x1
   4: ++(x1,nil()) -> x1
   5: hd(c(x1)) -> x1
   6: hd(++(c(x1),x2)) -> x1

 #2 -----------
   1: ++(++(x1,x2),x3) -> ++(x1,++(x2,x3))
   2: ++(x1,++(x2,x3)) -> ++(++(x1,x2),x3)
   3: ++(nil(),x1) -> x1
   4: ++(x1,nil()) -> x1
   7: *(nil(),x2) -> nil()
   8: *(s(x1),x2) -> ++(x2,*(x1,x2))

 #3 -----------
   9: q(q(x1,x2),x3) -> q(x1,q(x2,x3))
  10: q(x1,q(x2,x3)) -> q(q(x1,x2),x3)
  11: q(e(),x1) -> x1
  12: q(x1,e()) -> x1
  13: plus(x1,plus(x2,x3)) -> plus(plus(x1,x2),x3)
  14: plus(x1,x2) -> plus(x2,x1)
  15: plus(0(),x1) -> x1
  16: plus(s(x1),x2) -> s(plus(x1,x2))
  17: len(e()) -> 0()
  18: len(a()) -> s(0())
  19: len(q(a(),x2)) -> plus(len(a()),len(x2))

@Jouannaud and Kirchner's criterion
--- R
   1: ++(++(x1,x2),x3) -> ++(x1,++(x2,x3))
   2: ++(x1,++(x2,x3)) -> ++(++(x1,x2),x3)
   3: ++(nil(),x1) -> x1
   4: ++(x1,nil()) -> x1
   5: hd(c(x1)) -> x1
   6: hd(++(c(x1),x2)) -> x1

--- S
```

```
   1: ++(++(x1,x2),x3) -> ++(x1,++(x2,x3))
   2: ++(x1,++(x2,x3)) -> ++(++(x1,x2),x3)
   3: ++(nil(),x1) -> x1
   4: ++(x1,nil()) -> x1
   5: hd(c(x1)) -> x1
   6: hd(++(c(x1),x2)) -> x1


@Jouannaud and Kirchner's criterion
--- R
   1: ++(++(x1,x2),x3) -> ++(x1,++(x2,x3))
   2: ++(x1,++(x2,x3)) -> ++(++(x1,x2),x3)
   3: ++(nil(),x1) -> x1
   4: ++(x1,nil()) -> x1
   7: *(nil(),x2) -> nil()
   8: *(s(x1),x2) -> ++(x2,*(x1,x2))

--- S
   1: ++(++(x1,x2),x3) -> ++(x1,++(x2,x3))
   2: ++(x1,++(x2,x3)) -> ++(++(x1,x2),x3)
   3: ++(nil(),x1) -> x1
   4: ++(x1,nil()) -> x1
   7: *(nil(),x2) -> nil()
   8: *(s(x1),x2) -> ++(x2,*(x1,x2))


@Commutation Lemma

@Rule Labeling
--- R
   9: q(q(x1,x2),x3) -> q(x1,q(x2,x3))
  10: q(x1,q(x2,x3)) -> q(q(x1,x2),x3)
  11: q(e(),x1) -> x1
  12: q(x1,e()) -> x1
  13: plus(x1,plus(x2,x3)) -> plus(plus(x1,x2),x3)
  14: plus(x1,x2) -> plus(x2,x1)
  15: plus(0(),x1) -> x1
  16: plus(s(x1),x2) -> s(plus(x1,x2))
  17: len(e()) -> 0()
  18: len(a()) -> s(0())
  19: len(q(a(),x2)) -> plus(len(a()),len(x2))

--- S
```

```
  10: q(x1,q(x2,x3)) -> q(q(x1,x2),x3)

@Commutation Lemma

@Rule Labeling
--- R
  10: q(x1,q(x2,x3)) -> q(q(x1,x2),x3)

--- S
   9: q(q(x1,x2),x3) -> q(x1,q(x2,x3))
  11: q(e(),x1) -> x1
  12: q(x1,e()) -> x1
  13: plus(x1,plus(x2,x3)) -> plus(plus(x1,x2),x3)
  14: plus(x1,x2) -> plus(x2,x1)
  15: plus(0(),x1) -> x1
  16: plus(s(x1),x2) -> s(plus(x1,x2))
  17: len(e()) -> 0()
  18: len(a()) -> s(0())
  19: len(q(a(),x2)) -> plus(len(a()),len(x2))

@Jouannaud and Kirchner's criterion
--- R
   9: q(q(x1,x2),x3) -> q(x1,q(x2,x3))
  11: q(e(),x1) -> x1
  12: q(x1,e()) -> x1
  13: plus(x1,plus(x2,x3)) -> plus(plus(x1,x2),x3)
  14: plus(x1,x2) -> plus(x2,x1)
  15: plus(0(),x1) -> x1
  16: plus(s(x1),x2) -> s(plus(x1,x2))
  17: len(e()) -> 0()
  18: len(a()) -> s(0())
  19: len(q(a(),x2)) -> plus(len(a()),len(x2))

--- S
   9: q(q(x1,x2),x3) -> q(x1,q(x2,x3))
  11: q(e(),x1) -> x1
  12: q(x1,e()) -> x1
  13: plus(x1,plus(x2,x3)) -> plus(plus(x1,x2),x3)
  14: plus(x1,x2) -> plus(x2,x1)
  15: plus(0(),x1) -> x1
  16: plus(s(x1),x2) -> s(plus(x1,x2))
  17: len(e()) -> 0()
  18: len(a()) -> s(0())
```

```
19: len(q(a(),x2)) -> plus(len(a()),len(x2))
```

Listing B.5: Sample Input III.

```
(VAR x y z)
(RULES
  +(x,y)      -> +(y,x)
  +(+(x,y),z) -> +(x,+(y,z))
  +(0,x)      -> x
  +(s(x),y)   -> s(+(x,y))
  fib(x)      -> :(x,+(fib(x),fib(+(x,1))))
  *(x,y)      -> *(y,x)
  *(0,y)      -> 0
  *(s(x),y)   -> +(y,*(x,y))
  sq(x)       -> *(x,x)
  sq(s(x))    -> +(*(x,x),s(+(x,x)))
  .(.(x,y),z) -> .(x,.(y,z))
  .(x,.(y,z)) -> .(.(x,y),z)
  .(a,.(b,a)) -> .(.(b,a),b)
)
```

Listing B.6: Sample Output III.

```
NO

2 decompositions
 #1 -----------
  12: .(.(x2,x1),x3) -> .(x2,.(x1,x3))
  13: .(x2,.(x1,x3)) -> .(.(x2,x1),x3)
  14: .(a(),.(b(),a())) -> .(.(b(),a()),b())

 #2 -----------
   1: +(x2,x1) -> +(x1,x2)
   2: +(+(x2,x1),x3) -> +(x2,+(x1,x3))
   3: +(0(),x2) -> x2
   4: +(s(x2),x1) -> s(+(x2,x1))
   5: fib(x2) -> :(x2,+(fib(x2),fib(+(x2,1()))))
   6: fib(0()) -> 0()
   7: *(x2,x1) -> *(x1,x2)
   8: *(0(),x1) -> 0()
   9: *(s(x2),x1) -> +(x1,*(x2,x1))
  10: sq(x2) -> *(x2,x2)
  11: sq(s(x2)) -> +(*(x2,x2),s(+(x2,x2)))

unjoinable peak
    .(.(a(),b()),.(.(b(),a()),b()))
*<- .(.(a(),.(b(),a())),.(b(),a())) ->*
    .(.(.(b(),a()),b()),.(b(),a()))
```