Confluence of Prefix-Constrained Rewrite Systems

Nirina Andrianarivelo
LIFO - Université d'Orléans, B.P. 6759, 45067 Orléans cedex 2, France
Nirina.Andrianarivelo@univ-orleans.fr

Pierre Réty
LIFO - Université d'Orléans, B.P. 6759, 45067 Orléans cedex 2, France
Pierre.Rety@univ-orleans.fr

Abstract

Prefix-constrained rewriting is a strict extension of context-sensitive rewriting. We study the confluence of prefix-constrained rewrite systems, which are composed of rules of the form $L : l \to r$ where L is a regular string language that defines the allowed rewritable positions. The usual notion of Knuth-Bendix's critical pair needs to be extended using regular string languages, and the convergence of all critical pairs is not enough to ensure local confluence. Thanks to an additional restriction we get local confluence, and then confluence for terminating systems, which makes the word problem decidable. Moreover we present an extended Knuth-Bendix completion procedure, to transform a non-confluent prefix-constrained rewrite system into a confluent one.

2012 ACM Subject Classification Theory of computation → Rewrite systems

Keywords and phrases prefix-constrained term rewriting, confluence, critical pair

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.6

1 Introduction

Term rewriting is a rule-based formalism that can be used to study properties of functional programs, security protocols, musical rhythmics,... More generally, it provides a finite abstraction of a system whose configurations are represented by ranked terms. In this framework, and also to ensure the termination of rewrite computations, it is often necessary to restrict the possible rewrite positions, using strategies, or by allowing only some redex positions. In context-sensitive rewriting [10], some arguments of a function symbol may be defined as being non-rewritable. Prefix-constrained rewriting [8] is an extension of context-sensitive rewriting, where rewritable positions are defined by a regular string language that indicates the allowed prefixes.

Given a term t, a normal form of t is an irreducible term (denoted $t{\downarrow}$) obtained by rewriting t. Termination of a rewrite relation $\to_R$ ensures the existence of normal forms, whereas confluence ensures their uniqueness. Together, termination and confluence ensure that the word problem is decidable, because $t =_R t'$ is equivalent to $t{\downarrow} = t'{\downarrow}$. On the other hand, from a functional programming point of view, termination ensures that any program run will terminate, and confluence ensures that all functions are deterministic, i.e. each function call yields at most one result. These properties have also been addressed for context-sensitive rewriting ([5] for termination and [11] for confluence). On the other hand, the termination of prefix-constrained rewriting has been addressed in [1]. Both [5] and [1] consist in transforming the context-sensitive or prefix-constrained rewrite system into an ordinary one by a termination-preserving transformation, and studying the termination of the ordinary rewrite system.

In this paper, we study the confluence of prefix-constrained rewrite systems. In contrast to ordinary rewriting, prefix-constrained rewriting (and context-sensitive rewriting) is not closed under context application, which is a major difference. This is why the usual notion of Knuth-Bendix's critical pair needs to be extended (using regular string languages), and the convergence of all critical pairs is not enough to ensure local confluence. Thanks to an additional restriction we get local confluence, and then confluence for terminating systems, which makes the word problem decidable. Moreover we present an extended Knuth-Bendix completion procedure, to transform a non-confluent prefix-constrained rewrite system into a confluent one.

The paper is organized as follows. The preliminaries are introduced in Section 2. Local-confluence is studied in Section 3, and a comparison with [11] is given at the end of the section. The operational point of view to handle string languages is given in Section 4. An extended Knuth-Bendix completion procedure is presented in Section 5. Further work is outlined in Section 6.

2 Preliminaries

Term and Substitution. Consider a *finite ranked alphabet* Σ and a set of variables X. Each symbol $f \in \Sigma$ has a unique arity, denoted by $ar(f)$. The notions of *first-order term*, *position* and *substitution* are defined as usual. $T(\Sigma, X)$ denotes the set of terms over $\Sigma \cup X$, and $T(\Sigma)$ denotes the set of ground terms (without variables) over Σ. For a term t, $Var(t)$ is the set of variables of t, $Pos(t)$ is the set of positions of t, $PosVar(t)$ is the set of variable positions of t, $PosNonVar(t) = Pos(t) \backslash PosVar(t)$, and ϵ is the root position. For $p \in Pos(t)$, $t(p)$ is the symbol of $\Sigma \cup X$ occurring at position p in t, and $t|_p$ is the subterm of t at position p. For $p, p' \in Pos(t)$, $p < p'$ means that p occurs in t strictly above p', whereas $p \parallel p'$ means that $p \neq p'$ and $p \not< p'$ and $p' \not< p$. The term t is *linear* if each variable of t occurs only once in t. The term $t[t']_p$ is obtained from t by replacing the subterm at position p by t'.

Given σ and σ' two substitutions, $\sigma \circ \sigma'$ denotes the substitution such that for all variable x, $\sigma \circ \sigma'(x) = \sigma(\sigma'(x))$. The substitution σ is a *unifier* of the terms t and t' if $\sigma(t) = \sigma(t')$. If in addition, for all unifier θ of t and t', there exists a substitution γ such that $\theta = \gamma \circ \sigma$, then σ is called the *most general unifier* of t and t' (denoted $mgu(t, t')$). If it exists, the most general unifier is unique up to a variable renaming.

Term Rewrite System (TRS). A *rewrite rule* is an oriented pair of terms, written $l \to r$. We always assume that l is not a variable, and $Var(r) \subseteq Var(l)$. A *rewrite system* R is a finite set of rewrite rules. *lhs* stands for left-hand-side, *rhs* for right-hand-side. The *rewrite relation* $\to_R$ is defined as follows: $t \to_R t'$ if there exist a non-variable position $p \in Pos(t)$, a rule $l \to r \in R$, and a substitution θ s.t. $t|_p = \theta(l)$ and $t' = t[\theta(r)]_p$ (also denoted $t \to_R^p t'$). $\to_R^+$ denotes the transitive closure of $\to_R$, and $\to_R^*$ denotes the reflexive-transitive closure of $\to_R$. t' is a *descendant* of t if $t \to_R^* t'$. If I is a set of ground terms, $R^*(I)$ denotes the set of descendants of elements of I. The rewrite rule $l \to r$ is *left (resp. right) linear* if l (resp. r) is linear. R is *left (resp. right) linear* if all its rewrite rules are left (resp. right) linear. R is *linear* if R is both left and right linear. $l \to r$ is said *collapsing* if r is a variable.

Let $l_1 \to r_1$ and $l_2 \to r_2$ be rewrite rules such that $l_1|_p$ and l_2 are unifiable for some $p \in PosNonVar(l_1)$. Let $\sigma = mgu(l_1|_p, l_2)$. Then the pair of terms $(\sigma(r_1), \sigma(l_1)[\sigma(r_2)]_p)$ is called *critical pair*[1].

Context-Sensitive Term Rewrite System (CS-TRS) [4, 10]. A *context-sensitive rewrite relation* is a sub-relation of the ordinary rewrite relation in which rewritable positions

[1] As usual, we do not consider trivial critical pairs $(\sigma(r_1), \sigma(r_1))$ coming from the case where $l_1 = l_2$ and $r_1 = r_2$ and $p = \epsilon$.

are indicated by specifying arguments of function symbols. A mapping $\mu : \Sigma \to P(\mathbb{N})$ is said to be a *replacement map* (or Σ-map) if $\mu(f) \subseteq \{1, \dots, ar(f)\}$ for all $f \in \Sigma$. A *context-sensitive* term rewriting system (CS-TRS) is a pair $\mathcal{R} = (R, \mu)$ composed of a TRS and a replacement map. The set of μ-replacing positions[2] $Pos^{\mu}(t)$ ($\subseteq Pos(t)$) is recursively defined: $Pos^{\mu}(t) = \{\epsilon\}$ if t is a constant or a variable, otherwise $Pos^{\mu}(f(t_1, \dots, t_n)) = \{\epsilon\} \cup \{i.p \mid i \in \mu(f),\, p \in Pos^{\mu}(t_i)\}$. The rewrite relation induced by a CS-TRS $\mathcal{R}$ is defined: $t \hookrightarrow_{\mathcal{R}} t'$ if $t \to^{p}_{R} t'$ for some $p \in Pos^{\mu}(t)$.

▶ **Example 1.** Let $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$ and $R = \{a \to b\}$ with $\mu(f) = \{1\}$ and $\mu(g) = \{2\}$. The positions allowed by μ in the term $\boldsymbol{f}(\boldsymbol{a}, a)$ are written in bold. Then the only derivation issued from this term is $f(a, a) \hookrightarrow_{\mathcal{R}} f(b, a)$. On the other hand, consider $t = \boldsymbol{f}(\boldsymbol{g}(a, \boldsymbol{a}), a)$. Then the only derivation issued from this term is $f(g(a, a), a) \hookrightarrow_{\mathcal{R}} f(g(a, b), a)$.

String Language. Given an alphabet Σ, the set of all strings over Σ is denoted by Σ^*, and ϵ denotes the *empty string*. Symbol '.' denotes the concatenation.

String Automaton. A finite *string automaton* is a 5-tuple $\mathcal{A} = (\Sigma, Q, Q_I, Q_f, \Delta)$ where Q is a set of states, $Q_I \subseteq Q$ is the set of initial states, $Q_f \subseteq Q$ is the set of final states, and $\Delta \subseteq Q \times \Sigma \times Q$ is the set of *transitions*. The *transition relation* $\mapsto_{\Delta}$ between elements of $Q \times \Sigma^*$ is defined as follows: for $q, q' \in Q$, $a \in \Sigma$, $w \in \Sigma^*$, $(q, a.w) \mapsto_{\Delta} (q', w)$ iff $(q, a, q') \in \Delta$. The reflexive-transitive closure of $\mapsto_{\Delta}$ is written $\mapsto^*_{\Delta}$. The language recognized by $\mathcal{A}$ is $L_{\mathcal{A}} = \{w \in \Sigma^* \mid \exists q_I \in Q_I, \exists q_f \in Q_f, (q_I, w) \mapsto^*_{\Delta} (q_f, \epsilon)\}$. A *regular string language* is a set of strings recognized be some finite string automaton. It is well known that regular languages are closed under union, intersection, complement, and membership and emptiness are decidable.

$\mathcal{A}$ is said *deterministic* (resp. *complete*) if Q_I contains at most (resp. at least) one state, and for each $q \in Q$ and $a \in \Sigma$, there exists at most (resp. at least) one $q' \in Q$ such that $(q, a, q') \in \Delta$. It is well known that every automaton can be determinized and completed into an automaton that recognizes the same language. However the determinization step is exponential in the number of states. Let us write $\bar{\mathcal{A}} = (\Sigma, Q, Q_I, Q \backslash Q_f, \Delta)$. If $\mathcal{A}$ is deterministic and complete, it is well known that $\bar{\mathcal{A}}$ is deterministic and complete, and $L_{\bar{\mathcal{A}}} = \Sigma^* \backslash L_{\mathcal{A}}$, i.e. $\bar{\mathcal{A}}$ recognizes the complement of the language of $\mathcal{A}$.

Consider the automata $\mathcal{A}_1 = (\Sigma, Q^1, Q^1_I, Q^1_f, \Delta^1)$ and $\mathcal{A}_2 = (\Sigma, Q^2, Q^2_I, Q^2_f, \Delta^2)$. Let us define the automaton $\mathcal{A}_1 \cap \mathcal{A}_2 = (\Sigma,\, Q^1 \times Q^2,\, Q^1_I \times Q^2_I,\, Q^1_f \times Q^2_f,\, \Delta^1 \otimes \Delta^2)$ with $\Delta^1 \otimes \Delta^2 = \{((q_1, q_2), a, (q'_1, q'_2)) \mid (q_1, a, q'_1) \in \Delta_1 \wedge (q_2, a, q'_2) \in \Delta_2\}$. It is well known that $L_{\mathcal{A}_1 \cap \mathcal{A}_2} = L_{\mathcal{A}_1} \cap L_{\mathcal{A}_2}$, i.e. the automaton $\mathcal{A}_1 \cap \mathcal{A}_2$ recognizes the language intersection. Moreover if $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic and complete, so is $\mathcal{A}_1 \cap \mathcal{A}_2$.

Prefix Constrained Term Rewrite System (pCTRS) [8]. Prefix constrained rewriting allows rewrite steps only at the positions p of t s.t. the path from the root of t and p belongs to a given regular string language. More precisely, consider the set of directions $Dir(\Sigma) = \{\langle g, i\rangle \mid g \in \Sigma, 1 \leq i \leq ar(g)\}$. For a variable $x \in X$, let $path(x, \epsilon) = \epsilon$ and for a term $t = g(t_1, \dots, t_{(ar(g))}) \in T(\Sigma, X)$ and a position p, $path(t, p) \in Dir(\Sigma)^*$ is defined recursively by:

$$path(g(t_1, \dots, t_{(ar(g))}), \epsilon) = \epsilon$$
$$path(g(t_1, \dots, t_{(ar(g))}), i.p) = \langle g, i\rangle.path(t_i, p) \ \ \textit{with } 1 \leq i \leq ar(g) \text{ and } i.p \in Pos(t)$$

[2] Also called positions allowed by μ.

A *prefix constrained rewrite system* is a finite set R of prefix constrained rewrite rules of the form $L : l \to r$ s.t. $L \subseteq Dir(\Sigma)^*$ is a regular string language over $Dir(\Sigma)$, $l \in T(\Sigma, \mathcal{X})\backslash\mathcal{X}$, and $r \in T(\Sigma, var(l))$. A term t is rewritten to t' in one step by a pCTRS R, denoted by $t \hookrightarrow_R t'$, if there exist a prefix-constrained rewrite rule $L : l \to r$ in R, a position $p \in Pos(t)$ s.t. $path(t, p) \in L$, and a substitution σ s.t. $t|_p = \sigma(l)$ and $t' = t[\sigma(r)]_p$. The reflexive-transitive closure of $\hookrightarrow_R$ is denoted by $\hookrightarrow_R^*$. The equality $=_R$ is the reflexive, symmetric and transitive closure of the pCTRS rewriting $\hookrightarrow_R$. A rewrite step $t \hookrightarrow_R t'$ at position p of t by rewrite rule $L : l \to r$ through substitution σ is noted $t \hookrightarrow_{[p,L:l\to r,\sigma]} t'$. Let us note that prefix-constrained rewriting is stable under instantiation.

▶ **Example 2.** Let $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$ and $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$. Let $t = \boldsymbol{f}(g(a, \boldsymbol{a}), a)$. Note that $t(1.2) = a$ (in bold in t) and $path(t, 1.2) = \langle f,1\rangle.\langle g,2\rangle \in (\langle f,1\rangle.\langle g,2\rangle)^*$. Then this position can be reduced by prefix constrained rewriting, i.e. $t = \boldsymbol{f}(g(a, \boldsymbol{a}), a) \hookrightarrow_R \boldsymbol{f}(g(a, b), a)$, whereas the other occurrences of a are not reducible. Note that the term $f(a, a)$ is not reducible by the pCTRS R, whereas it is reducible by the CS-TRS of Example 1. However the pCTRS $R_1 = \{(\langle f,1\rangle|\langle g,2\rangle)^* : a \to b\}$ is equivalent to the CS-TRS of Example 1.

▶ Remark. Context-sensitive rewriting is a particular case of prefix-constrained rewriting [8].

Confluence and Church-Rosser Property. For any binary relation S over the set of terms, let S^* be the reflexive-transitive closure of S, and $=_S$ be the reflexive-symmetric-transitive closure of S.
We say that the pair of terms (t_1, t_2) *converges* for S (denoted $t_1 \downarrow_S t_2$) if there exists a term t' such that $t_1\, S^*\, t'$ and $t_2\, S^*\, t'$.
S is said *locally confluent* if $t\, S\, t_1$ and $t\, S\, t_2$ implies $t_1 \downarrow_S t_2$, for all terms t, t_1, t_2.
S is said *confluent* if $t\, S^*\, t_1$ and $t\, S^*\, t_2$ implies $t_1 \downarrow_S t_2$, for all terms t, t_1, t_2.
S has the *Church-Rosser property* if $t_1 =_S t_2$ implies $t_1 \downarrow_S t_2$, for all terms t, t_1, t_2.

▶ **Theorem 3.** *[3] Church-Rosser property and confluence are equivalent.*

S is said *terminating* (or *well-founded*) if there is no infinite sequence of terms $t_1\, S\, t_2\, S\, t_3\, S\, \ldots$.

▶ **Theorem 4.** *(Newman's lemma) [6] If S is locally confluent and terminating, then S is confluent.*

Now, let us consider a TRS R and the associated binary relation $\to_R$.

▶ **Theorem 5.** *(Knuth-Bendix's theorem) [9] R is locally confluent*[3] *if and only if all critical pairs of R are convergent.*

3 Local Confluence of pCTRSs

When positions p_1 and p_2 are parallel, rewriting at p_1 does not change the prefix of p_2, and conversely. Therefore such a peak converges as for ordinary TRSs.

▶ **Lemma 6.** *Let $R = \{L_1 : l_1 \to r_1,\ L_2 : l_2 \to r_2\} \cup R'$ be a pCTRS.*
If $t \hookrightarrow_{[p_1,L_1:l_1\to r_1,\sigma_1]} t_1$ and $t \hookrightarrow_{[p_2,L_2:l_2\to r_2,\sigma_2]} t_2$ and $p_1 \parallel p_2$,
then $t_1 \hookrightarrow_{[p_2,L_2:l_2\to r_2,\sigma_2]} t_3$ and $t_2 \hookrightarrow_{[p_1,L_1:l_1\to r_1,\sigma_1]} t_3$.

[3] I.e. $\to_R$ is locally confluent.

Proof. Since $p_1||p_2$, we have $t_1|_{p_2} = t|_{p_2}$ and $path(t_1, p_2) = path(t, p_2) \in L_2$. Then $t_1 \hookrightarrow t_3$. Since $p_1||p_2$, we have $t_2|_{p_1} = t|_{p_1}$ and $path(t_1, p_1) = path(t, p_1) \in L_1$. Then $t_2 \hookrightarrow t_3$. ◀

With an ordinary TRS, a peak coming from an overlap in a variable position converges. It may be wrong when considering a pCTRS. Consequently, a pCTRS without critical pairs may not be locally confluent.

▶ **Example 7.** Consider the pCTRS $R = \{\{\epsilon\} : f(x) \to g(x), \{\langle f, 1\rangle\} : a \to b\}$. So $f(a) \hookrightarrow_R g(a)$ and $f(a) \hookrightarrow_R f(b) \hookrightarrow_R g(b)$. Note that $g(a)$ is irreducible by R because the second rewrite rule needs a prefix with symbol f to be applied. Then this peak starting from $f(a)$ does not converge, therefore R is not locally confluent. Moreover R does not have critical pairs.

In the previous example, the non-convergent peak comes from the fact that along the step $f(a) \hookrightarrow_R g(a)$, the occurrence of a in $f(a)$ is allowed to be reduced by the second rule, whereas it is forbidden in $g(a)$. In other words, the prefix $\langle f, 1\rangle$ of a in $f(a)$ belongs to the language of the second rule, whereas the prefix $\langle g, 1\rangle$ of a in $g(a)$ does not. We introduce the notion of *prefix-preserving* to avoid this situation, which is based on the same idea as in the context-sensitive case (Definition 4.4 of [11]).
Notations : for a variable x and a term t, let $Pos(t, x) = \{p \in Pos(t) \mid t(p) = x\}$. On the other hand, we use the character '.' to denote the string concatenation.

▶ **Definition 8.** The pCTRS R is *prefix-preserving* if for all rewrite rules $L_1 : l_1 \to r_1$ and $L_2 : l_2 \to r_2$ of R, for all $x \in Var(l_1)$, for all $p, p' \in Pos(l_1, x)$, for all $p'' \in Pos(r_1, x)$, for all $u, w \in Dir(\Sigma)^*$:

$$u \in L_1 \wedge u.path(l_1, p).w \in L_2 \implies u.path(l_1, p').w \in L_2 \wedge u.path(r_1, p'').w \in L_2$$

In the previous definition, p' is for considering the case where l_1 is not linear. The pCTRS of Example 7 is not prefix-preserving (with $u = w = \epsilon$).

▶ **Example 9.** Consider the pCTRS $R = \{L : \mathit{if}(true, x, y) \to x,\ L : \mathit{if}(false, x, y) \to y\}$, where L is the set of all words of $Dir(\Sigma)^*$ except the words that contain at least one occurrence of $\langle \mathit{if}, 2\rangle$ or $\langle \mathit{if}, 3\rangle$. Thus, the first argument (the condition) of *if* should be evaluated before the second or the third argument. R is prefix-preserving because the position of x (resp. y) in the left-hand-side of the first (resp. second) rule is forbidden with respect to L, i.e. the pre-condition of the implication of Definition 8, that means more precisely, $\forall u \in Dir(\Sigma)^*, \forall w \in Dir(\Sigma)^*\ \ u.path(l, pos(l, x)).w \in L$, is always wrong.

▶ **Lemma 10.** *Let $R = \{L_1 : l_1 \to r_1,\ L_2 : l_2 \to r_2\} \cup R'$ be a prefix-preserving pCTRS.
If $t \hookrightarrow_{[p_1, L_1 : l_1 \to r_1, \sigma_1]} t_1$ and $t \hookrightarrow_{[p_2, L_2 : l_2 \to r_2, \sigma_2]} t_2$ and $p_2 = p_1.v.w$ with $v \in Pos(l_1, x)$ for some variable x,
then there exist t_3 and t_4 such that $t_1 \hookrightarrow^*_{[p_1.v'_1.w, \ldots, p_1.v'_m.w, L_2 : l_2 \to r_2, \sigma_2]} t_4$
and $t_2 \hookrightarrow^*_{[p_1.v_1.w, \ldots, p_1.v_n.w, L_2 : l_2 \to r_2, \sigma_2]} t_3 \hookrightarrow_{[p_1, L_1 : l_1 \to r_1, \sigma'_1]} t_4$,
with $Pos(r_1, x) = \{v'_1, \ldots, v'_m\}$ and $Pos(l_1, x) = \{v, v_1, \ldots, v_n\}$.*

Proof. Let us write $u = path(t, p_1)$. Then we have $u \in L_1$ and $path(t, p_1.v.w) \in L_2$. But $path(t, p_2) = path(t_1, p_1.v.w) = u.path(l_1, v).path(\sigma_1(x), w) \in L_2$.
Since R is prefix-preserving, we have $\forall i, u.path(l_1, v_i).path(\sigma_1(x), w) \in L_2$, then $t_2 \to^* t_3$ and $u.path(r_1, v'_i).path(\sigma_1(x), w) \in L_2$, then $t_1 \to^* t_4$.
Furthermore, $path(t_3, p_1) = path(t, p_1) \in L_1$ then $t_3 \hookrightarrow_{[p_1, L_1 : l_1 \to r_1, \sigma'_1]} t_4$ with $\sigma'_1(x) = \sigma_1(x)[\sigma_2(r_2)]_w$ and $\forall y,$ s.t. $y \neq x, \sigma'_1(y) = \sigma_1(y)$. ◀

Prefix-preserving is helpful to get local confluence, but it is not always necessary. The following pCTRS is locally confluent, whereas it is not prefix-preserving.

▶ **Example 11.**

$$R = \{\{\epsilon\} : f(x) \xrightarrow{1} g(x),\ \{\langle f,1\rangle.\langle h,1\rangle\} : a \xrightarrow{2} b,\ \{\langle g,1\rangle\} : h(a) \xrightarrow{3} h(b)\}$$

The only peak is $f(h(a)) \hookrightarrow g(h(a))$ by rule 1, and $f(h(a)) \hookrightarrow f(h(b))$ by rule 2. This peak is convergent since $g(h(a)) \hookrightarrow g(h(b))$ by rule 3 and $f(h(b)) \hookrightarrow g(h(b))$ by rule 1. Therefore R is locally confluent, and consequently confluent since R is terminating. Note the use of rule number 3 to get confluence.
Let L_1 and L_2 be the prefix-languages of rules 1 and 2 respectively. R is not prefix-preserving because using the notations of Definition 8, let $u = \epsilon \in L_1$ and $w = \langle h,1\rangle$, and we have $u.path(f(x),1).w = \langle f,1\rangle.\langle h,1\rangle \in L_2$ whereas $u.path(g(x),1).w = \langle g,1\rangle.\langle h,1\rangle \notin L_2$.
If we replace rule 3 by $\{\langle g,1\rangle\} : h(x) \xrightarrow{3'} h(b)$, the pCTRS is not terminating anymore, but it is still locally confluent and not prefix-preserving.

As seen above, the prefix-constraints of a pCTRS could be annoying to get local confluence. However, they could also be favorable.

▶ **Example 12.** The TRS $R = \{f(a) \to c,\ a \to b\}$ is not locally confluent because $f(a) \to_R c$, $f(a) \to_R f(b)$, and c and $f(b)$ are irreducible. Actually there is a critical pair $(c, f(b))$, which is not convergent.

Now, Consider the pCTRS $R' = \{\{\epsilon\} : f(a) \to c,\ \{\epsilon\} : a \to b\}$. Now there is only one derivation issued from $f(a)$, i.e. $f(a) \hookrightarrow_{R'} c$, because the occurrence of a in $f(a)$ is forbidden for the second rule. Actually, the pCTRS R' is locally confluent, and the previous critical pair $(c, f(b))$ is not relevant for R'.

The definition of critical pairs should be modified to fit pCTRSs.

▶ **Definition 13.** (critical pair for a pCTRS) Let $L_1 : l_1 \to r_1$ and $L_2 : l_2 \to r_2$ be prefix-constrained rewrite rules such that $l_1|_p$ and l_2 are unifiable for $\forall p \in PosNonVar(l_1)$. Let $\sigma = mgu(l_1|_p, l_2)$ and $L = \{u \in L_1 \mid u.path(l_1, p) \in L_2\}$. If $L \neq \emptyset$, the triple $(\sigma(r_1),\ \sigma(l_1)[\sigma(r_2)]_p,\ L)$ is called a *critical pair*.

Let us notice that L is necessarily regular.

When considering the rules of the pCTRS R' of Example 12, we get $p = 1$ and $L = \emptyset$. Therefore the critical pair $(c, f(b))$ of the TRS R does not produce a critical pair for the pCTRS R'.

If there is a peak coming from an overlap at a non-variable position, then there is a critical pair.

▶ **Lemma 14.** *(extended critical pair lemma)*
Let $R = \{L_1 : l_1 \to r_1,\ L_2 : l_2 \to r_2\} \cup R'$ be a pCTRS.
If $t \hookrightarrow_{[p_1, L_1 : l_1 \to r_1, \sigma_1]} t_1$ and $t \hookrightarrow_{[p_2, L_2 : l_2 \to r_2, \sigma_2]} t_2$ and $p_2 = p_1.v$ with $v \in PosNonVar(l_1)$, then there exists a critical pair (s_1, s_2, L) and a substitution γ such that $path(t, p_1) \in L$ and $t_1 = t[\gamma(s_1)]_{p_1}$ and $t_2 = t[\gamma(s_2)]_{p_1}$.

Proof. Let us assume $Var(l_1) \cap Var(l_2) = \emptyset$. Since $t \hookrightarrow t_1$, we have $path(t, p_1) \in L_1$ and $t|_{p_1} = \sigma_1(l_1)$. Since $t \hookrightarrow t_2$, we have $path(t, p_2) \in L_2$ and $t|_{p_2} = \sigma_2(l_2)$. Then $\sigma_2(l_2) = t|_{p_2} = (t|_{p_1})|_v = (\sigma_1(l_1))|v = (\sigma_1(l_1|_v))$ because $v \in PosNonVar(l_1)$. Let us write $\theta = \sigma_1 \cup \sigma_2$. Then $l_1|_v$ and l_2 are unifiable by θ and there exists a substitution γ s.t

$\theta = \gamma \circ mgu(l|_v, l_2)$. Let us write $L = \{u \in L_1 \mid u.path(l_1, v) \in L_2\}$ and $\alpha = mgu(l_1|_v, l_2)$. Then $path(t, p_1) \in L$ because $path(t, p_1) \in L_1$ and $path(t, p_2) = path(t, p_1).path(l_1, v) \in L_2$. Then $L \neq \emptyset$, consequently $(\alpha(r_1), (\alpha(l_1)[\alpha(r_2)]_v, L)$ is a critical pair. Let us write $s_1 = \alpha(r_1)$ and $s_2 = \alpha(l_1)[\alpha(r_2)]_v$. Then $t[\gamma(s_1)]_{p_1} = t[\gamma(\alpha(r_1))]_{p_1} = t[\theta(r_1)]_v = t[\sigma_1(r_1)]_v = t_1$. Moreover $t[\gamma(s_2)]_{p_1} = t[\gamma(\alpha(l_1))[\gamma(\alpha(r_2))]_v]_{p_1} = t[\theta(l_1)[\theta(r_2)]_v]_{p_1} = t[\sigma_1(l_1)[\sigma_2(r_2)]_v]_{p_1} = t[\sigma_2(r_2)]_{p_1.v} = t[\sigma_2(r_2)]_{p_2} = t_2$ ◀

Conversely, if there is a critical pair, then there is a peak.

▶ **Lemma 15.** *Let $L_1 : l_1 \to r_1$ and $L_2 : l_2 \to r_2$ be prefix-constrained rewrite rules. If $(\sigma(r_1), \sigma(l_1)[\sigma(r_2)]_v, L)$ is a critical pair, then for each term t and $p_1 \in Pos(t)$:*

$$t|_{p_1} = \sigma(l_1) \wedge path(t, p_1) \in L \implies t \hookrightarrow_{[p_1, L_1 : l_1 \to r_1]} t[\sigma(r_1)]_{p_1} \wedge t \hookrightarrow_{[p_1.v, L_2 : l_2 \to r_2]} t[\sigma(r_2)]_{p_1.v}$$

Note that at least one pair (t, p_1) exists since $L \neq \emptyset$.

Proof. Since $L \subseteq L_1$ then $path(t, p_1) \in L_1$ therefore $t \hookrightarrow_{[p_1, L_1 : l_1 \to r_1]} t[\sigma(r_1)]_{p_1}$. On the other hand, $path(t, p_1.v) = path(t, p_1).path(t|_{p_1}, v) = path(v, p_1).path(l_1, v)$. Moreover, $path(t, p_1) \in L$, consequently $path(t, p_1.v) \in L_2$. $t|_{p_1.v} = (t|_{p_1})|_v = (\sigma(l_1))|_v = \sigma(l_1|_v) = \sigma(l_2)$. Therefore $t \hookrightarrow_{[p_1.v, L_2 : l_2 \to r_2]} t[\sigma(r_2)]_{p_1.v}$ ◀

▶ **Definition 16.** The critical pair (s_1, s_2, L) is said *convergent* if

$$\forall t \in T(\Sigma, X), \forall p \in Pos(t),\ (path(t, p) \in L \implies t[s_1]_p \downarrow_R t[s_2]_p)$$

▶ **Theorem 17.** *(extended Knuth-Bendix's theorem) Let R be a prefix-preserving pCTRS. R is locally confluent if and only if all critical pairs of R are convergent.*

Proof.

1. "$\Longrightarrow$ "
   Let us write $s_1 = \sigma(r_1)$ and $s_2 = \sigma(l_1[\sigma(r_2)]_v$ and $t' = t[\sigma(l_1)]_p$. Through Lemma 15 applied on t' and p, we get $t' \hookrightarrow t'[s_1]_p = t[s_1]_p$ and $t' \hookrightarrow t'[\sigma(r_2)]_{p.v} = t[\sigma(l_1)[\sigma(r_2)]_v]_p = t[s_2]_p$. Through the local confluence property, $t[s_1]_p \downarrow_R t[s_2]_p$.
2. " $\Longleftarrow$"
   Assume $t \hookrightarrow_{[p_1, L1 : l_1 \to r_1, \sigma_1]} t_1$ and $t \hookrightarrow_{[p_2, L2 : l_2 \to r_2, \sigma_2]} t_2$
   - if $p_1 || p_2$, through Lemma 6, $t_1 \hookrightarrow t_3 \hookleftarrow t_2$
   - without loss of generality, assume $p_1 < p_2$
     - if $p_2.p_1 \notin PosNonVar(l_1)$, through Lemma 10, we have $t_1 \hookrightarrow^* t_4 \hookleftarrow^* t_2$.
     - otherwise through lemma 14, there exist a substitution γ, and a critical pair (s_1, s_2, L), s.t. $path(t, p_1) \in L$ and $t_1 = t[\gamma(s_1)]_{p_1}$ and $t_2 = [\gamma(s_2)]_{p_1}$. This critical pair is convergent and since $path(t, p_1) \in L$, we have $t[s_1]_{p_1} \hookrightarrow^* \ \hookleftarrow^* t[s_2]_{p_1}$. Since, pCTRS rewriting is stable through instantiation, $t_1 \hookrightarrow^* \ \hookleftarrow^* t_2$

◀

In general, to check the convergence of a critical pair according to Definition 16, infinitely many contexts t should be tried, which is impossible. Therefore we need to define a stronger sufficient condition. Let us first introduce the notion of rewriting under a prefix language. As usual, for a string w and a string language L, we define $L.w$ by $L.w = \{v.w \mid v \in L\}$.

▶ **Definition 18.** Let $R = \{L : l \to r\} \cup R'$ be a pCTRS, and $L' \subseteq Dir(\Sigma)^*$.
$t \overset{L'}{\hookrightarrow}_{[p,L:l\to r,\sigma]} t'$ if $L'.path(t,p) \subseteq L$ and σ is a substitution s.t. $t|_p = \sigma(l)$ and $t' = t[\sigma(r)]_p$.
We also write $t \overset{L'}{\hookrightarrow}_R t'$, and $\overset{L'}{\hookrightarrow}^*_R$ will denote the reflexive-transitive closure of $\overset{L'}{\hookrightarrow}_R$.

▶ Remark. $t \hookrightarrow_{[p,L:l\to r]} t' \iff t \overset{\{\epsilon\}}{\hookrightarrow}_{[p,L:l\to r]} t'$.

▶ **Lemma 19.** *If* $t \overset{L'}{\hookrightarrow}_{[p,L:l\to r]} t'$ *then*

$$\forall t_0 \in T(\Sigma, X), \forall p' \in Pos(t_0), (path(t_0, p') \in L' \implies t_0[t]_{p'} \hookrightarrow_{[p'.p,L:l\to r]} t_0[t']_{p'})$$

Proof. Through the hypothesis $t \overset{L'}{\hookrightarrow}_{[p,L:l\to r]} t'$, we have $t \to_{[p,l\to r]} t'$ and $L'.path(t,p) \subseteq L$.
Assume $path(t_0, p') \in L'$. Then $path(t_0[t]_{p'}, p'.p) = path(t_0, p').path(t, p) \in L$. Then $t_0[t]_{p'} \hookrightarrow_{[p'.p,L:l\to r]} t_0[t']_{p'}$. ◀

▶ **Corollary 20.** *If* $L' \neq \emptyset$ *and* $t_1 \overset{L'}{\hookrightarrow}_R t_2 \overset{L'}{\hookrightarrow}_R \cdots \overset{L'}{\hookrightarrow}_R t_n$,
then there exists $t_0 \in T(\Sigma, X)$ *and* $p' \in Pos(t_0)$ *s.t.* $t_0[t_1]_{p'} \hookrightarrow_R t_0[t_2]_{p'} \hookrightarrow_R \cdots \hookrightarrow_R t_0[t_n]_{p'}$.

▶ **Corollary 21.** *If* $L' \neq \emptyset$ *and* $\overset{L'}{\hookrightarrow}_R$ *is not terminating, then* $\hookrightarrow_R$ *is not terminating.*
Consequently, if $\hookrightarrow_R$ *is terminating, then* $\overset{L'}{\hookrightarrow}_R$ *is terminating.*

▶ **Definition 22.** The critical pair (s_1, s_2, L) is said *strongly convergent* if there exits a term t such that $s_1 \overset{L}{\hookrightarrow}^*_R t$ and $s_2 \overset{L}{\hookrightarrow}^*_R t$.

Therefore, if the pCTRS R is terminating, the strong convergence of a critical pair (s_1, s_2, L) can be checked by computing all descendants of s_1 and of s_2 under prefix-language L, which are finitely many, and looking for common elements.

▶ **Lemma 23.** *Strong convergence implies convergence.*

Proof. We have $s_1 \overset{L}{\hookrightarrow}^* s_3$ and $s_2 \overset{L}{\hookrightarrow}^* s_3$. Let $t_0 \in T(\Sigma), p \in Pos(t_0)$ s.t $path(t_0, p) \in L$.
Through corollary 20
$t_0[s_1]_p \hookrightarrow^* t_0[s_3]_p$ and $t_0[s_2]_p \hookrightarrow^* t_0[s_3]_p$ then the critical pair is convergent. ◀

▶ **Theorem 24.** *Let* R *be a prefix-preserving pCTRS.*
If all critical pairs of R *are strongly convergent, then* R *is locally confluent.*

Proof. It is naturally deduced from Lemma 23 and Theorem 17. ◀

Let us note that the converse is wrong as illustrated by the following Example .

▶ **Example 25.** Consider the pCTRS

$$R = \{\{\langle h,1\rangle\} : f(a) \to c,\ \{\langle h,1\rangle.\langle f,1\rangle\} : a \to b,\ \{\epsilon\} : h(f(b)) \to h(c)\}$$

R is prefix-preserving since the rewrite rules do not contain variables.
There is only one critical pair $(c,\ f(b),\ \{\langle h,1\rangle\})$, which is convergent because $h(f(b)) \hookrightarrow_R h(c)$.
From Theorem 17, R is locally confluent. However, the critical pair is not strongly convergent since according to Definition 18, c and $f(b)$ are irreducible by $\overset{\{\langle h,1\rangle\}}{\hookrightarrow}_R$.

The context-sensitive case

In this section we compare the previous results with those of [11]. A CS-TRS (R_0, μ) may be viewed as a particular pCTRS $R = \{L_k : l_k \to r_k \mid (l_k \to r_k) \in R_0\}$, where all languages L_k are the same (say L), and L is composed of all strings (including the empty string) over the alphabet $\{\langle f, i\rangle \mid f \in \Sigma, i \in \mu(f)\}$.

▶ **Example 26.** $R_0 = \{f(h(x, y), y) \to g(x, y), h(a, b) \to i(a), a \to b\}$ s.t. $\mu(f) = \mu(h) = 1$,
then we can view this CS-TRS as the following pCTRS

$$R = \{L : f(h(x, y), y) \to g(x, y), L : h(a, a) \to i(a), L : a \to b\} \text{ s.t. } L = (\langle f, 1\rangle \mid \langle h, 1\rangle)^*.$$

In this framework, note that $u, v, w \in L \iff u.v.w \in L$, and $path(t, p) \in L \iff p \in Pos^\mu(t)$. Consequently, the pCTRS R is prefix-preserving if and only if the CS-TRS (R_0, μ) is with left homogeneous replacing variables (Definition 4.4 of [11]).

According to Definition 13, a critical pair between $L : l_1 \to r_1$ and $L : l_2 \to r_2$ is of the form $(\sigma(r_1), \sigma(l_1)[\sigma(r_2)]_p, L')$ where $L' = \{u \in L \mid u.path(l_1, p) \in L\} \neq \emptyset$. Since $L' \neq \emptyset$, there exists at least one string $u \in L$ s.t. $u.path(l_1, p) \in L$. Therefore $path(l_1, p) \in L$, then $p \in Pos^\mu(l_1)$. On the other hand, for all $u \in L$, we have $u.path(l_1, p) \in L$ (because $path(l_1, p) \in L$). Consequently $L' = L$.

For instance, for the example 26 we have two critical pairs

$$(f(i(a), a)), g(a, a), L)$$
$$(h(b, a), i(a), L).$$

Since all critical pairs have the same language, we can omit it, and we get the same notion (called μ-critical pair) as in Definition 4.7 of [11].

Therefore, Theorem 17 gives the same result as Theorem 4.9 of [11]. However, as mentioned previously, with a pCTRS infinitely many contexts should be tried to check the convergence of a critical pair. Fortunately $\epsilon \in L$, and when using Definition 16 we can consider p as the root position, i.e. the critical pair should also converge without context. Conversely, if the critical pair converges without context, it also converges with any context t assuming $p \in Pos^\mu(t)$, which holds because $path(t, p) \in L$ is assumed. Thus, the convergence of a critical pair can be checked without using a context, i.e. as in [11].

As a conclusion, if the pCTRS is context-sensitive, thanks to Theorem 17 we get the same result as [11]. If the pCTRS is not context-sensitive, we can ensure local-confluence using Theorem 24.

4 Working with String Automata

From an operational point of view, Section 3 does not say anything for handling prefix languages, for checking whether a pCTRS is prefix-preserving, for computing the language of a critical pair, and for computing $\overset{L}{\hookrightarrow}_R$ steps. This is why we consider in this section that for a pCTRS $R = \{L_k : l_k \to r_k \mid 1 \leq k \leq n\}$, each language $L_k \subseteq Dir(\Sigma)^*$ is defined by a finite string automaton $\mathcal{A}^k = (Dir(\Sigma), Q^k, Q^k_I, Q^k_f, \Delta^k)$.

For efficiency of further computations, we assume that each each $\mathcal{A}^k$ is deterministic and complete. In other words, we consider that the automata are determinized and completed at the beginning, and the new automata generated by the computations will be still deterministic and complete.

For any automaton $\mathcal{A} = (\Sigma, Q, Q_I, Q_f, \Delta)$ and $q \in Q$, let us define $L_{\mathcal{A}}(q) = \{w \in \Sigma^* \mid \exists q_f \in Q_f,\ (q, w) \mapsto^*_{\Delta} (q_f, \epsilon)\}$. Note that $L_{\mathcal{A}} = \cup_{q_I \in Q_I} L_{\mathcal{A}}(q_I)$, and $L_{\mathcal{A}}(q)$ is recognized by the automaton $(\Sigma, Q, \{q\}, Q_f, \Delta)$.

Let w be a string and $L \subseteq \Sigma^*$ be a string language. Let us define $L^{w-} = \{u \in \Sigma^* \mid w.u \in L\}$ and $L^{-w} = \{u \in \Sigma^* \mid u.w \in L\}$.

For a string w, let $Q_I^{w-} = \{q \in Q \mid \exists q_I \in Q_I, (q_I, w) \mapsto^*_{\Delta} (q, \epsilon)\}$, and let $Q_f^{-w} = \{q \in Q \mid \exists q_f \in Q_f, (q, w) \mapsto^*_{\Delta} (q_f, \epsilon)\}$. Let $\mathcal{A}^{w-} = (\Sigma, Q, Q_I^{w-}, Q_f, \Delta)$ and $\mathcal{A}^{-w} = (\Sigma, Q, Q_I, Q_f^{-w}, \Delta)$.

▶ **Lemma 27.** *$L_{\mathcal{A}^{w-}} = (L_{\mathcal{A}})^{w-}$ and $L_{\mathcal{A}^{-w}} = (L_{\mathcal{A}})^{-w}$. Moreover, if $\mathcal{A}$ is deterministic and complete, so are $\mathcal{A}^{w-}$ and $\mathcal{A}^{-w}$.*

Proof.

1. Let us prove that $L_{\mathcal{A}^{w-}} \subseteq (L_{\mathcal{A}})^{w-}$
   Let $u \in L_{\mathcal{A}^{w-}}$. There exists $q \in Q_I^{w-}$ and $q_f \in Q_f$ such that $(q, u) \mapsto^*_{\Delta} (q_f, \epsilon)$. Then there exists $q_I \in Q_I$ such that $(q_I, w) \mapsto^*_{\Delta} (q, \epsilon)$. Consequently, $(q_I, w.u) \mapsto^*_{\Delta} (q, u) \mapsto^*_{\Delta} (q_f, \epsilon)$. Then $w.u \in L_{\mathcal{A}}$, then $u \in (L_{\mathcal{A}})^{w-}$.
2. Let us prove $(L_{\mathcal{A}})^{w-} \subseteq L_{\mathcal{A}^{w-}}$
   Let $u \in (L_{\mathcal{A}})^{w-}$. Then $w.u \in L_{\mathcal{A}}$. Consequently, there exists $q_I \in Q_I$, $q_f \in Q_f$ and $q \in Q$ such that $(q_I, w.u) \mapsto^*_{\Delta} (q, u) \mapsto^*_{\Delta} (q_f, \epsilon)$. Then $(q_I, w) \mapsto^*_{\Delta} (q, \epsilon)$, that is $q \in Q_I^{w-}$. Consequently, $u \in (L_{\mathcal{A}})^{w-}$.
3. If $\mathcal{A}$ is deterministic and complete, then $|Q_I^{w-}| = |\{q \in Q \mid \exists q_I \in Q_I, (q_I, w) \mapsto^*_{\Delta} (q, \epsilon)\}|$, where $|A|$ denotes the number of elements of the set A, as usual. But $|Q_I| = 1$ and Δ is the set of the transitions in $\mathcal{A}$. Then $|Q_I^{w-}| = 1$. On the other part, the transitions in $\mathcal{A}^{w-}$ and $\mathcal{A}$ are the same.
4. Let us prove $L_{\mathcal{A}^{-w}} \subseteq (L_{\mathcal{A}})^{-w}$
   Let $u \in L_{\mathcal{A}^{-w}}$. There exists $q_I \in Q_I$, $q \in Q_f^{-w}$ such that $(q_I, u) \mapsto^*_{\Delta} (q, \epsilon)$. Then there exists $q_f \in Q_f$, $(q, w) \mapsto^*_{\Delta} (q_f, \epsilon)$. Consequently, $(q_I, u.w) \mapsto^*_{\Delta} (q, w) \mapsto^*_{\Delta} (q_f, \epsilon)$. Then $u.w \in L_{\mathcal{A}}$, the $u \in (L_{\mathcal{A}})^{-w}$.
5. $(L_{\mathcal{A}})^{-w} \subseteq L_{\mathcal{A}^{-w}}$
   Let $u \in (L_{\mathcal{A}})^{-w}$. Then $u.w \in L_{\mathcal{A}}$. Consequently, there exists $q_I \in Q_I$, $q_f \in Q_f$ and $q \in Q$ such that $(q_I, u.w) \mapsto^*_{\Delta} (q, w) \mapsto^*_{\Delta} (q_f, \epsilon)$. Then $q \in Q_f^{-w}$ and $(q_I, u) \mapsto^*_{\Delta} (q, \epsilon)$. Consequently, $u \in L_{\mathcal{A}^{-w}}$.
6. The initial states and the transitions of $\mathcal{A}^{-w}$ and $\mathcal{A}$ are the same.

◀

Prefix preserving. To check whether a pCTRS is prefix-preserving (Definition 8), we use the following result.

▶ **Theorem 28.** *For the prefix-constrained rewrite rules $L_1 : l_1 \to r_1$ and $L_2 : l_2 \to r_2$, let $\mathcal{A}_1 = (\Sigma, Q^1, Q^1_I, Q^1_f, \Delta^1)$ and $\mathcal{A}_2 = (\Sigma, Q^2, Q^2_I, Q^2_f, \Delta^2)$ be deterministic and complete automata that recognize L_1 and L_2 respectively. Let $S_{1,2} =$ $\{q \in Q^2 \mid \exists (q^1_I, q^2_I) \in Q^1_I \times Q^2_I, \exists u \in Dir(\Sigma)^*, \exists q^1_f \in Q^1_f, ((q^1_I, q^2_I), u) \mapsto^*_{\Delta^1 \otimes \Delta^2} ((q^1_f, q), \epsilon)\}$, which can be computed by saturating $\{(q^1_I, q^2_I)\}$ with the transitions of $\Delta^1 \otimes \Delta^2$.*

The pCTRS R is prefix-preserving if and only if for all rewrite rules $L_1 : l_1 \to r_1$ and $L_2 : l_2 \to r_2$ of R, $\forall x \in Var(l_1)$, $\forall p, p' \in Pos(l_1, x)$, $\forall p'' \in Pos(r_1, x)$, $\forall q \in S_{1,2}$:

$$L_{\mathcal{A}_2}(q)^{path(l_1,p)-} = L_{\mathcal{A}_2}(q)^{path(l_1,p')-} \subseteq L_{\mathcal{A}_2}(q)^{path(r_1,p'')-}$$

Proof.

- Let us begin by proving the implication "$\Longrightarrow$".
  Let us assume $u \in L_1$ and $u.path(l_1, p).w \in L_2$. Then there exists $q_I^1 \in Q_I^1$, $q_I^2 \in Q_I^2, q_f^1 \in Q_f^1, q \in Q^2$ such that $(q_I^1, u) \rightarrow^*_{\Delta^1} (q_f^1, \epsilon)$ and $(q_I^2, u) \rightarrow^*_{\Delta^2} (q, \epsilon)$. Then $q \in S_{1,2}$. Moreover, $path(l_1, p).w \in L_{\mathcal{A}_2}(q)$, then $w \in L_{\mathcal{A}_2}(q)^{path(l_1,p)-}$. Through the hypothesis, we have $w \in L_{\mathcal{A}_2}(q)^{path(l_1,p')-}$ and $w \in L_{\mathcal{A}_2}(q)^{path(l_1,p'')-}$. Then $path(l_1, p').w \in L_{\mathcal{A}_2}(q)$ and $path(r_1, p'').w \in L_{\mathcal{A}_2}(q)$. As $(q_I^2, u) \rightarrow^*_{\Delta^2} (q, \epsilon)$, we get $u.path(l_1, p').w \in L_{\mathcal{A}_2} = L_2$ and $u.path(r_1, p'').w \in L_{\mathcal{A}_2} = L_2$. Consequently, R is prefix-preserving.
- Let us continue by proving the implication "$\Longleftarrow$".
  Let $q \in S_{1,2}$ and $w \in L_{\mathcal{A}_2}(q)^{path(l_1,p)-}$. There exists $u \in Dir(\Sigma)^*$, $q_I^1 \in Q_I^1$, $q_I^2 \in Q_I^2, q_f^1 \in Q_f^1$ such that $(q_I^1, u) \rightarrow^*_{\Delta^1} (q_f^1, \epsilon)$ and $(q_I^2, u) \rightarrow^*_{\Delta^2} (q, \epsilon)$. Then $u \in L_1$. Moreover $path(l_1, p).w \in L_{\mathcal{A}_2}(q)$, then there exists $q_f^2 \in Q_f^2$ such that $(q, path(l_1, p).w) \rightarrow^*_{\Delta^2} (q_f^2, \epsilon)$. Consequently, $u.path(l_1, p).w \in L_2$. As R is prefix-preserving, we have $u.path(l_1, p').w \in L_2 = L_{\mathcal{A}_2}$ and $u.path(r_1, p'').w \in L_2 = L_{\mathcal{A}_2}$. Then $path(l_1, p').w \in L_{\mathcal{A}_2}(q)$ and $path(r_1, p'').w \in L_{\mathcal{A}_2}(q)$, then $w \in L_{\mathcal{A}_2}(q)^{path(l_1,p')-}$ and $w \in L_{\mathcal{A}_2}(q)^{path(r_1,p'')-}$. Therefore, $L_{\mathcal{A}_2}(q)^{path(l_1,p)-} \subseteq L_{\mathcal{A}_2}(q)^{path(l_1,p')-}$ and $L_{\mathcal{A}_2}(q)^{path(l_1,p)-} \subseteq L_{\mathcal{A}_2}(q)^{path(r_1,p'')-}$. On the other hand, we prove that $L_{\mathcal{A}_2}(q)^{path(l_1,p')-} \subseteq L_{\mathcal{A}_2}(q)^{path(l_1,p)-}$ by exchanging p and p'.

◀

Critical pair. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two deterministic and complete automata that recognize the languages L_1 and L_2 of the two rules in the critical pair Definition 13. Then the language L of the critical pair is recognized by the automaton $\mathcal{A}_1 \cap \mathcal{A}_2^{-path(l_1,p)}$, which is still deterministic and complete.

Rewriting under context. Let $\mathcal{A}'$ and $\mathcal{A}$ be deterministic and complete automata that recognizes the languages L' and L of Definition 18. To check whether $L'.path(t, p) \subseteq L$, we use the following result:

▶ **Lemma 29.** $L'.path(t, p) \subseteq L \iff L_{(\mathcal{A}' \cap (\bar{\mathcal{A}})^{-path(t,p)})} = \emptyset$.

Proof.

1. "$\Longleftarrow$"
   By contradiction. Let us assume there exists $u \in L'$ such that $u.path(t, p) \notin L$. Then $u.path(t, p) \in \bar{L} = L_{\bar{\mathcal{A}}}$. Then $u \in (L_{\bar{\mathcal{A}}})^{-path(t,p)} = L_{(\bar{\mathcal{A}})^{-path(t,p)}}$. Since $u \in L'$, we have $u \in L_{\mathcal{A}'}$, then $u \in L_{\mathcal{A}' \cap (\bar{\mathcal{A}})^{-path(t,p)}} = \emptyset$ according to the hypothesis. Contradiction.
2. "$\Longrightarrow$"
   By contradiction. Let assume there exists $u \in L_{\mathcal{A}' \cap (\bar{\mathcal{A}})^{-path(t,p)}}$. Then $u \in L'$ and $u.path(t, p) \in \bar{L}$, that is $u.path(t, p) \notin L$ and $u.path(t, p) \in L'.path(t, p)$. Consequently, $L'.path(t, p) \not\subseteq L$. Contradiction.

◀

Note that checking equalities or inclusions of languages as in Theorem 28, or computing the complement as in Lemma 29, is polynomial since automata are deterministic and complete.

5 Extended Knuth-Bendix Completion

The goal of extended completion is to transform an arbitrary initial pCTRS R (or a set of equalities) into a confluent and terminating pCTRS R' without changing the equality modulo the pCTRS, i.e. such that $=_R$ and $=_{R'}$ are identical. To do it, we use the result of

Theorem 24, therefore the pCTRS needs to be prefix-preserving. However, with the notations of Definition 8, whenever $u \in L_1 \wedge u.path(l_1, p).w \in L_2$ whereas $u.path(r_1, p'').w \notin L_2$, i.e. the pCTRS is not prefix-preserving, we could make it prefix-preserving by extending L_2 into L_2' so that $u.path(r_1, p'').w \in L_2'$. Unfortunately, this may change the equality modulo the pCTRS.

▶ **Example 30.** Let $\Sigma = \{f, g, a, b, c, d\}$ and

$$R = \{\{\epsilon\} : f(x) \xrightarrow{1} g(x, c),\ \{\langle f, 1\rangle\} : a \xrightarrow{2} b\}$$

Let L_1 and L_2 be the prefix-languages of rules 1 and 2 respectively. R is not prefix-preserving because, let $u = \epsilon \in L_1$ and $w = \epsilon$, and we have $u.path(f(x), 1).w = \langle f, 1\rangle \in L_2$ whereas $u.path(g(x, c), 1).w = \langle g, 1\rangle \notin L_2$. Note that R is not confluent since there is the peak $f(a) \hookrightarrow g(a, c)$ by rule 1, and $f(a) \hookrightarrow f(b) \hookrightarrow g(b, c)$ by rules 1 and 2, which is not convergent since $g(a, c)$ and $g(b, c)$ are irreducible.
Now let us extend L_2 by considering the pCTRS:

$$R' = \{\{\epsilon\} : f(x) \xrightarrow{1} g(x, c),\ \{\langle f, 1\rangle\} \cup \{\langle g, 1\rangle\} : a \xrightarrow{2'} b\}$$

R' is prefix-preserving. However $g(a, d) \hookrightarrow_{R'} g(b, d)$ whereas $g(a, d) \neq_R g(b, d)$. In other words, $=_R$ and $=_{R'}$ are not identical.

This difficulty may depend on the orientation of rewrite rules. The pCTRS R of Example 30 is terminating, however let us reverse the first rule of R, i.e. let

$$R'' = \{\{\epsilon\} : g(x, c) \xrightarrow{1''} f(x),\ \{\langle f, 1\rangle\} : a \xrightarrow{2} b\}$$

R'' is prefix-preserving and is also terminating. Moreover $=_R$ and $=_{R''}$ are identical. Unfortunately, changing the orientation does not always make the pCTRS prefix-preserving, and may not preserve termination. In other words, extended completion will fail when one cannot get a prefix-preserving pCTRS.

The usual Knuth-Bendix completion generates an inter-reduced TRS R, which means (roughly) that the left-hand-side and the right-hand-side of each rule of R are not reducible by the other rules of R. This notion cannot be extended to pCTRSs in an easy way.

▶ **Example 31.** Consider the TRS $R = \{f(x) \xrightarrow{1} g(x), g(x) \xrightarrow{2} h(x)\}$. Then R is not inter-reduced since the right-hand-side of rule 1 is reducible by rule 2. Now let

$$R' = \{\{\langle i, 1\rangle\} \cup \{\langle j, 1\rangle\} : f(x) \xrightarrow{1'} g(x),\ \{\langle i, 1\rangle\} : g(x) \xrightarrow{2'} h(x)\}$$

So, the right-hand-side of rule 1 is reducible by rule 2 under context i, but not under context j. An inter-reduced pCTRS R'' such that $=_{R'}$ and $=_{R''}$ are identical, could be

$$R'' = \{\{\langle i, 1\rangle\} : f(x) \xrightarrow{1''} h(x),\ \{\langle j, 1\rangle\} : f(x) \xrightarrow{1'''} g(x),\ \{\langle i, 1\rangle\} : g(x) \xrightarrow{2''} h(x)\}$$

In this paper, we present a basic extended Knuth-Bendix completion, which does not attempt to produce an inter-reduced pCTRS. It is described by inference rules, as in [2], and computes (P_{i+1}, R_{i+1}) from (P_i, R_i) using a derivation relation denoted $\vdash$, where P_i, P_{i+1} are sets of prefix-constrained equalities of the form $L : p = q$[4] and R_i, R_{i+1} are sets of prefix-constrained rewrite rules of the form $L : l \to r$.

[4] We assume that $=$ is commutative, i.e. $L : p = q$ is the same equality as $L : q = p$.

1. **Orient**

$$\frac{P \cup \{L : p = q\},\ R}{P,\ R \cup \{L : p \to q\}} \quad \text{if } R \cup \{L : p \to q\} \text{ is prefix-preserving and terminating}$$

2. **Deduce**

$$\frac{P,\ R}{P \cup \{L : p = q\},\ R} \quad \text{if } (p, q, L) \text{ is a critical pair between rules of } R$$

3. **Simplify**

$$\frac{P \cup \{L : p = q\},\ R}{P \cup \{L : p' = q\},\ R} \quad \text{if } p \overset{L}{\hookrightarrow}_R p'$$

4. **Delete**

$$\frac{P \cup \{L : p = p\},\ R}{P,\ R}$$

Orient needs to check that $R \cup \{L : p \to q\}$ is terminating. This can be done by transforming the pCTRS into an ordinary TRS [1], which preserves termination, and checking the termination of the ordinary TRS using the usual techniques and tools. This transformation can even be done incrementally: each time *Orient* is run, new rewrite rules are added into the ordinary TRS.

With our basic completion above, inference rules *Simplify* and *Delete* are only applied on (non-oriented) equations of P. During the completion procedure, oriented rules of R are neither simplified nor deleted.

▶ **Lemma 32.** *(Soundness) If $(P, R) \vdash (P', R')$, then $=_{P \cup R}$ and $=_{P' \cup R'}$ are identical.*

Proof.

1. *Orient* :
$t =_{L:p=q} t' \iff t =_{L:p\to q} t'$ because $=_{L:p=q}$ is a symmetric relation.
2. *Deduce* :
Let us consider the critical pair (p, q, L) obtained from the rewriting rules $L_1 : l_1 \to r_1 \in R$ and $L_2 : l_2 \to r_2 \in R$. Then $(p, q, L) = (\sigma(r_1), \sigma(l_1)[\sigma(r_2)]_v, L)$. If $t' =_{[p',L:p=q,\theta]} t''$, then $t' = t_0[\theta(p)]_{p'}, t'' = t_0[\theta q]_{p'}$ and $path(t_0, p') \in L$. Let us write $t = t_0[\sigma(l_1)]_{p'}$. Through Lemma 15, we have $t \hookrightarrow_{[p',L_1:l_1\to r_1]} t[\sigma(r_1)]_{p'} = t_0[p]_{p'}$ and $t \hookrightarrow_{[p'.v,L_2:l_2\to r_2]} t[\sigma(r_2)]_{p'.v} = t_0[\sigma(l_1[\sigma(r_2)]_v]_{p'}) = t_0[q]_{p'}$. Since $t' =_{[p',L:p=q,\theta]} t''$, let us assume $Var(p) \cap Var(t') = \emptyset$ and $Var(q) \cap Var(t'') = \emptyset$. Then $Var(p) \cap Var(t_0) = \emptyset$ and $Var(q) \cap Var(t_0) = \emptyset$. Consequently, $\theta(t) \hookrightarrow_{[p',L_1:l_1\to r_1]} t_0[\theta(p)]_{p'} = t'$ and $\theta(t) \hookrightarrow_{[p'.v,L_2:l_2\to r_2]} t_0[\theta(q)]_{p'} = t''$. Then $t' =_R t''$.
3. *Simplify*
   - **a.** "$\Longrightarrow$ "
   If $t =_{[u,L:p=q,\sigma]} t'$, then we have $t|_u = \sigma(p)]$ and $t' = t[\sigma(q)]$ and $path(t, u) \in L$. But $p \overset{L}{\hookrightarrow}_{[v,L':l'\to r',\theta]} p'$, then $p|_v = \theta(l')$ and $p' = p[\theta(r')]_v$ and $L.path(p, v) \subseteq L'$. Consequently, $t \hookrightarrow_{[u.v,L':l'\to r',\sigma\theta]} t[\sigma(\theta(p')]_{u.v} = t[t|_u[\sigma(\theta(r')]_v]_u = t[(\sigma(p)[\sigma(\theta(r'))]_v]_u = t[\sigma(p([\theta(r'))]_v]_u = t[\sigma(p')]_u$ since $path(t, u.v) = path(t, u).path(p, v) \in L'$ (note that $path(t, u) \in L$). Furthermore, $t[\sigma(p')]_u =_{[L,p'=q]} t[\sigma(q)]_u = t'$ since $path(t, u) \in L$. Consequently $t =_{R\cup\{L:p'=q\}} t'$.
   - **b.** "$\Longleftarrow$ "
   The converse is similar since the direction of the rewrite step $p \hookrightarrow p'$ does not matter.

4. *Delete*
   If $t =_{L:p=p} t'$, then $t = t'$. Thus $t =_{P \cup R} t'$. ◀

Consider a derivation $(P_0, R_0) \vdash \cdots \vdash (P_n, R_n)$. Note that $R_0 \subseteq R_1 \subseteq \cdots \subseteq R_n$.

▶ **Lemma 33.** *(Completeness) Let (p, q, L) be a critical pair between some rules of R_n. If $p \overset{L}{\hookrightarrow}{}^*_{R_n} p'$ and $q \overset{L}{\hookrightarrow}{}^*_{R_n} q'$, then (p, q, L) is strongly convergent in $R_n \cup \{L : p' \to q'\}$.*

Proof. $p' \overset{L}{\hookrightarrow}_{[\epsilon, p' \to q']} q'$ since $L.path(p', \epsilon) = L \subseteq L$. Consequently, $p \overset{L}{\hookrightarrow}{}^*_{R_n} p' \overset{L}{\hookrightarrow}_{[\epsilon, L:p' \to q']} q'$ and $q \overset{L}{\hookrightarrow}{}^*_{R_n} q'$. Then the critical pair is strongly convergent in $R_n \cup \{L : p' \to q'\}$. ◀

Fairness hypothesis. $(P_0, R_0) \vdash \cdots \vdash (P_n, R_n)$ is *fair* if for all critical pair (p, q, L) between rules of R_n, there is some $i \in \{0, \ldots, n\}$ such that $(L : p = q) \in P_i$. In other words, all critical pairs have been computed thanks to *Deduce*. From Lemmas 32, 33 and Theorems 24, 4 we get:

▶ **Corollary 34.** *If $(P_0, R_0) \vdash \cdots \vdash (P_n, R_n)$ is fair and $R_0 = P_n = \emptyset$, then R_n is confluent and terminating. Moreover the relations $=_{P_0}$ and $=_{R_n}$ are identical.*

However, like the usual Knuth-Bendix completion, the extended Knuth-Bendix completion fails if we cannot obtain $P_n = \emptyset$ for some n. In particular, it arises if *Orient* cannot orient a persistent critical pair because the resulting pCTRS would not be prefix-preserving or would not be terminating.

The above basic completion could be improved by including more inference rules like *Simplifiying* and *Deleting* oriented rules of R. However, the proof of correctness and completeness of such completion procedure would be more complicated and could be done, for instance, by extending the proof transformation method of [2].

6 Conclusion and Further Work

In this paper, we present a sufficient condition that ensures the local confluence of prefix-constrained rewrite systems, and consequently the confluence of terminating ones. This result subsumes that of [11] about local-confluence of context-sensitive rewrite systems. Prefix-preserving and critical-pair strong convergence assumptions are sufficient, but are not necessary. Finding weaker assumptions is an interesting challenge.

The second contribution of this paper is an extended Knuth-Bendix completion procedure for prefix-constrained rewrite systems. This procedure could be improved to get inter-reduced systems, by adding some inference rules, which could also improve the efficiency.

Controlled rewriting [7] is an extension of prefix-constrained rewriting, where rewritable positions are defined by a regular tree language that considers the entire term (i.e. not only prefixes). It could be interesting to study local-confluence, and define a completion procedure for controlled rewrite systems.

References

1 Nirina Andrianarivelo, Vivien Pelletier, and Pierre Réty. Transforming prefix-constrained or controlled rewrite systems. In Mohamed Mosbah and Michaël Rusinowitch, editors, *SCSS 2017, The 8th International Symposium on Symbolic Computation in Software Science 2017, April 6-9, 2017, Gammarth, Tunisia*, volume 45 of *EPiC Series in Computing*, pages 49–62. EasyChair, 2017. URL: `http://www.easychair.org/publications/paper/Transforming_Prefix-constrained_or_Controlled_Rewrite_Systems`.

2 Leo Bachmair and Nachum Dershowitz. Completion for rewriting modulo a congruence. In Pierre Lescanne, editor, *Rewriting Techniques and Applications, 2nd International Conference, RTA-87, Bordeaux, France, May 25-27, 1987, Proceedings*, volume 256 of *Lecture Notes in Computer Science*, pages 192–203. Springer, 1987. doi:10.1007/3-540-17220-3_17.

3 Alonzo Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936. URL: http://www.jstor.org/stable/1989762.

4 Kokichi Futatsugi, Joseph A. Goguen, Jean-Pierre Jouannaud, and José Meseguer. Principles of OBJ2. In Mary S. Van Deusen, Zvi Galil, and Brian K. Reid, editors, *Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages, New Orleans, Louisiana, USA, January 1985*, pages 52–66. ACM Press, 1985. doi:10.1145/318593.318610.

5 Jürgen Giesl and Aart Middeldorp. Transformation techniques for context-sensitive rewrite systems. *J. Funct. Program.*, 14(4):379–427, 2004. doi:10.1017/S0956796803004945.

6 Gérard P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980. doi:10.1145/322217.322230.

7 Florent Jacquemard, Yoshiharu Kojima, and Masahiko Sakai. Controlled term rewriting. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Frontiers of Combining Systems, 8th International Symposium, FroCoS 2011, Saarbrücken, Germany, October 5-7, 2011. Proceedings*, volume 6989 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2011. doi:10.1007/978-3-642-24364-6_13.

8 Florent Jacquemard, Yoshiharu Kojima, and Masahiko Sakai. Term rewriting with prefix context constraints and bottom-up strategies. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2015. doi:10.1007/978-3-319-21401-6_9.

9 D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning 2: Classical Papers on Computational Logic 1967-1970*, pages 342–376. Springer, Berlin, Heidelberg, 1983.

10 S. Lucas. Context-Sensitive Computations in Functional and Functional logic Programs. *Journal of Functional and Logic Programming*, 1998(1), January 1998.

11 Salvador Lucas. Context-sensitive computations in confluent programs. In Herbert Kuchen and S. Doaitse Swierstra, editors, *Programming Languages: Implementations, Logics, and Programs, 8th International Symposium, PLILP'96, Aachen, Germany, September 24-27, 1996, Proceedings*, volume 1140 of *Lecture Notes in Computer Science*, pages 408–422. Springer, 1996. doi:10.1007/3-540-61756-6_100.