ACADEMIC PRESS

Available at www.ComputerScienceWeb.com

POWERED BY SCIENCE DIRECT®

Information and Computation 183 (2003) 245–274

Information and Computation

www.elsevier.com/locate/ic

The context-splittable normal form for Church–Rosser language systems

Jens R. Woinowski[1]

TU Darmstadt, Fachbereich Informatik, Wilhelminenstr. 7, 64283 Darmstadt, Germany

Abstract

In this paper the *context-splittable* normal form for rewriting systems defining Church–Rosser languages is introduced. Context-splittable rewriting rules look like rules of context-sensitive grammars with swapped sides. To be more precise, they have the form $uvw \to uxw$ with u, v, w being words, v being nonempty and x being a single letter or the empty word. It is proved that this normal form can be achieved for each Church–Rosser language and that the construction is effective. Some interesting consequences of this characterization are given, too.

Keywords: Church–Rosser languages; String-rewriting; Confluence; Growing context-sensitive languages

1. Introduction

Church–Rosser languages (CRLs) basically are defined by a *length-reducing* string rewriting system and a mechanism to handle the word ends [13]. Here we call such a defining system a *Church–Rosser language system* (CRLS). CRL is a very interesting class of languages for three reasons: (i) Their word problem can be decided in deterministic linear time, although they are a strict superset of the *deterministic context-free languages* (DCFL) [13]. (ii) Despite that fact, their definition is more intuitive than that of DCFL. (iii) They are the deterministic variant of the *growing context-sensitive languages* (GCSL), which was proved in [14] (for the definition of GCSL see [8]). Therefore, they fit into the Chomsky hierarchy very well [1,7,12].

By assigning weights to single letters one can also define a *weight function* for words. This is the basis of the following important characterization result about Church–Rosser languages:

E-mail addresses: jens.woinowski@sdm.de, woinowski@iti.informatik.tu-darmstadt.de.

[1] Address at time of first submission. New address is: sd&m AG, software design and management, Munich.

0890-5401/03/$ - see front matter
doi:10.1016/S0890-5401(03)00035-X

Allowing *weight-reduction* instead of length-reduction does not improve the expressive power [14]. The proof of Niemann and Otto includes an effective construction: Given a rewriting system with weight-reducing rules defining a Church–Rosser language it is possible to construct an equivalent one that only has length-reducing rules.

In this paper, we use this fact to show the effective existence of a *context-splittable normal form* for every Church–Rosser language L. The defining (length-reducing) rewriting system of L can be simulated by a weight-reducing system that has rules of the form $uvw \to uxw$ with u, v, w being words, v being nonempty and x being a single letter or the empty word. We do not use the term 'context-sensitive' in order to stress the fact that the two forms are not fully corresponding to each other. Because context-splittable rule can also be deleting (i.e. of the form $v \to \square$), we do not always get a context-sensitive rule by swapping the sides.

One consequence of this normal form result is that the information flow during reductions is underlying stronger restrictions in a *context-splittable CRLS* (csCRLS): Any movement of a letter in either direction needs at least as many rule applications as the distance to be accomplished. Although this is only a refinement of the linear time bound for the reductions in CRLSs, it might be handy for proofs.

As mentioned above, Niemann and Otto [14] proved that CRL are the deterministic variant of GCSL. The term "growing *context-sensitive* language" is somewhat misleading, since weight-increasing grammars are not context-sensitive in the sense of [7]: they are only *monotone* grammars. The term is only used for historical reasons, it was coined by Dahlhaus and Warmuth [8]. In this article we give a characterization of CRL which, when transferred to GCSL, justifies to call them context-sensitive.

This article is organized in the following way: In Section 2 we give some basic definitions. In Section 3 a definition for csCRLSs is given and our main result is stated. The proof that csCRLSs are a normal form for CRLSs is given by a construction described in Section 4, introducing the "weight-spreading technique." In Section 5 we prove the correctness of the resulting csCRLSs, including the proof of confluence and weight-reduction. In Section 6 we discuss consequences of the characterization of CRLs by csCRLSs. These include results about the relation between CRL and GCSL, about a normal form for automata, and about derivation graphs. We conclude this article with a section in which we shortly discuss why this result is not simply a normal form, but a characterization theorem. Appendix A contains the case distinctions which are necessary to complete the proof. We extracted them from the proof for the purpose of better readability.

Early states of these research results have been published in a technical report [16], and an extended abstract was presented in [17], which is improved here especially w.r.t. the question of confluence.

2. Basic definitions

2.1. String rewriting

The reader is assumed to be familiar with definitions and notations of confluent string rewriting. For details, see [6,10,13].

A *string-rewriting system* (*or simply rewriting system*) R on Σ is a subset of $\Sigma^* \times \Sigma^*$. For $(u,v) \in R$ we also write $(u \to v) \in R$ and call (u,v) a rule.

A string-rewriting system R on Σ is *confluent* if and only if for all $w,x,y \in \Sigma^*, w \to^* x$ and $w \to^* y$ imply that there exists $z \in \Sigma^*$ with $x \to^* z$ and $y \to^* z$.

Theorem 2.1 ([13], Corollary to Theorem 1.1). *If in a string-rewriting system R no left-hand side of a rule properly overlaps with the left-hand side of another rule or itself, no left-hand side is a proper substring of another left-hand side, and no two rules have identical left-hand sides but different right-hand sides, then R is confluent.*

A string-rewriting system R on Σ is called *normalized* if the following conditions hold for each rule $(u,v) \in R$:

1. $u \in I_{RR}(R \setminus \{(u,v)\})$ and
2. $v \in I_{RR}(R)$.

That means, the right-hand side v of each rule is irreducible and the left-hand side u can be reduced only by the rule (u,v) itself.

Note that in a normalized system no two rules $(u,v_1),(u,v_2)$ with $v_1 \neq v_2$ can exist.

A *weight function* is a function $f : \Sigma \to \mathbf{N}$. It is recursively extended to a function on Σ^* by $f(wa) := f(w) + f(a)$ and $f(\Box) := 0$ (where $\Box$ is the empty word) with $w \in \Sigma^*$, $a \in \Sigma$. An example for a weight function is the *length function* with $f(a) := 1$ for all $a \in \Sigma$, then $f(w) = |w|$.

Throughout this article, a string-rewriting system R is called a *weight-reducing system*, if there exists a weight function f such that $f(u) > f(v)$ for all $(u,v) \in R$.

2.2. Church–Rosser languages

Definition 1. A *Church–Rosser language system (CRLS)* is a 6-tuple $C = (\Gamma, \Sigma, R, t_l, l_r, Y)$ with a finite alphabet Γ, a terminal alphabet $\Sigma \subset \Gamma$ ($\Gamma \setminus \Sigma$ is the alphabet of nonterminals), a finite confluent weight-reducing system $R \subseteq \Gamma^* \times \Gamma^*$, left and right end marker words $t_l, k_r \in (\Gamma \setminus \Sigma)^* \cap I_{RR}(R)$, and an accepting letter $Y \in (\Gamma \setminus \Sigma) \cap I_{RR}(R)$. The *language defined by C* is defined as: $L_C := \{w \in \Sigma^* | t_l \cdot w \cdot k_r \to_R^* Y\}$.

A language L is called a *Church–Rosser language (CRL)* if there exists a CRLS C with $L_C = L$.

The definition of Church–Rosser languages is due to McNaughton et al. [13]. The definition of Church–Rosser language systems given here is a convenient notation for their definition, where it is also used that the expressive power of Church–Rosser languages is not enhanced by allowing arbitrary weight functions instead of the length function. This was proved by Niemann and Otto in [14].

Throughout the text we will use letters from the beginning of the roman alphabet to denote single letters. Variables for words will be letters from t onwards.

Example 1. The word $w \in \Sigma^*$ with length $|w| = n$ consists of the letters a_1 to a_n in Σ, which is denoted as $w = a_1 \cdots a_n$.

If we split a word into substrings, we usually use the same letter which we use for the word and add an index. For example, let $w = abcd$. Then we would use $w_1 = ab$, $w_2 = cd$ to address the two halves of w.

2.3. Automata accepting Church–Rosser languages

We do not give a formal definition of the automata for the language family CRL, the sDTPDAs. Despite that, it will be helpful to understand parts of the construction to give an informal definition of a very similar model which has a mode of operation that is closer to our construction.

These automata work as follows (cf. [3]): They have two stacks and a scanning head. The initial configuration is a left-hand stack with the left-hand end marker word [2] in it, its leftmost letter at the bottom of the stack. Initially, the right-hand stack contains the input, the leftmost letter of the input being on top, and below it the right-hand end marker word.

Now, letters from the right-hand stack are shifted to the left, until a suffix of the left-hand stack is the left-hand side of a reduction rule of the CRLS. This suffix is deleted from the left-hand stack and the right-hand side of the rule is pushed onto the right-hand stack.

Because of the confluence of the underlying string-rewriting system we can make this automaton deterministic by choosing only one rule for every left-hand side and by using the rule with the shortest left-hand side if two or more different left-hand sides apply. If the rewriting system is normalized, this is not necessary, because in any given situation at most one reduction rule can be applied.

The procedure is repeated until the right-hand stack is empty and the left-hand stack is irreducible. If the left-hand stack only contains the accepting letter of the CRLS the input is accepted. Note that since after a reduction operation the left-hand stack will always be irreducible, one can directly combine one shift with each reduce operation, as in [13].

3. The normal form

Definition 2. A CRLS $C = (\Gamma, \Sigma, R, ¢, \$, Y)$ is *context-splittable* (*we say C is a CSCRLS*) if $¢, \$, Y \in I_{RR}(R) \cap \Gamma \setminus \Sigma$ (let the *inner alphabet* be $\Gamma_{\text{inner}} := \Gamma \setminus \{¢, \$, Y\}$) and for any rule $r \in R$ there exists a splitting (u, v, w, x) with:

1. $r = (uvw, uxw)$.
2. v is nonempty: $v \in \Gamma^+$.
3. uvw may contain at most one $¢$ and if so at its beginning. Also it may contain at most one \$ which only may appear at the end. All other letters of uvw have to be from the inner alphabet Γ_{inner}:

   $$uvw \in \{¢, \square\} \cdot \Gamma_{\text{inner}}^* \cdot \{\$, \square\}.$$

[2] Note that both end marker words are irreducible.

4. x is a single letter not equal to ¢ or \$ or it is the empty word:

$$x \in \Gamma_{\text{inner}} \cup \{Y, \square\}.$$

5. If v contains ¢ or \$, then $x = Y$, u and w are empty, and v begins with ¢ and ends with \$:

$$v \in ¢ \cdot \Gamma^*_{\text{inner}} \cup \Gamma^*_{\text{inner}} \cdot \$ \Rightarrow v \in ¢ \cdot \Gamma^*_{\text{inner}} \cdot \$ \wedge u = w = \square \wedge x = Y.$$

6. If $x = Y$, then u and w are empty, and v begins with ¢ and ends with \$:

$$x = Y \Rightarrow v \in ¢ \cdot \Gamma^*_{\text{inner}} \cdot \$ \wedge u = w = \square.$$

The splitting (u, v, w, x) of a rule r allowed by this is called a *context-splitting*, u and w are called the *left and right context*.

Remark 1. It is obvious that $x = \square$ is only necessary if $u \in \{\square, ¢\}$ and $w \in \{\square, \$\}$ cannot be avoided, since otherwise one could transfer a letter of u or w to x.

Example 2. These are some examples for the meaning of the definition (the splittings are marked by dots):

- $ab \cdot dea \cdot ab \to ab \cdot a \cdot ab$,
- $ab \cdot de \cdot aab \to ab \cdot \square \cdot aab$ is another splitting of the same rule,
- $¢ \cdot ab \cdot \$ \to ¢ \cdot \square \cdot \$$,
- $¢ \cdot ab\$ \cdot \square \to ¢ \cdot \$ \cdot \square$ is *not a valid* splitting,
- $abc \to \square$ is a *deleting* context-splittable rule, and
- $abcd \to dcba$ is a rule that is not context-splittable.

Remark 2. We want to distinguish the notion of context-splittable CRLSs from context-sensitive grammars, because the former uses reductions and the latter productions for defining languages. Especially deleting rules of the form $v \to \square$ have no counterpart in context-sensitive grammars. Therefore we do not use the term context-sensitive. Note that all other deleting rules with $u \notin \{\square, ¢\}$ or $w \notin \{\square, \$\}$ can be split in a different way such that x is not the empty word, just as in the example given above.

Theorem 3.1. *Let C be a CRLS with language L_C. Then there exists a csCRLS C' with $L_{C'} = L_C$.*

Proof. We will give an effective construction for such a new csCRLS C'. □

4. Constructing a context-splittable CRLS

In this section we introduce our construction. First, we give an outline of its principles and some preliminaries. Then we show how we are simulating an automaton accepting, which is the main part of the construction. We also describe the main cases which we distinguish and give an example. The full case distinction can be found in Appendix A.

4.1. The construction principles

Let $C = (\Gamma, \Sigma, R, t_l, t_r, Y)$ be a CRLS. Without restriction of generality we may assume that R is length-reducing [14]. Furthermore, we can assume that R is normalized: In a confluent rewriting system, rules that have a left-hand side which can be reduced by another rule can be dropped, and reducing right-hand sides does not cause a conflict with length-reduction.

In order to construct a csCRLS C', we will use the following four principles:

1. Analogous to the automaton model our new system will have the property that during the whole reduction process there is always exactly one place in the word where the next reduction rule can be applied.
2. We will use a compression alphabet which can store more than one letter of the input (respectively, the derivated words) in one letter. This information will be represented by subscripts of the compression letters.
3. These compression letters will be enriched by surplus letters in their subscripts in order to spread necessary weight-reductions over more than one letter.
4. Rules of the original system will, in most cases, be simulated by three or four rules in the new system.

The confluent weight-reducing system will be built of five parts R_1 to R_5.

4.2. Construction preliminaries

Definition 3. With Γ being the alphabet of C which consists of all terminal and nonterminal letters, let $\overline{\Gamma}$ be a new alphabet, which is a disjoint copy of Γ. Then $^-$ denotes the bijective morphism that maps Γ into $\overline{\Gamma}$, e.g. a to $\overline{a}$. Let $\sharp$, $\not\sharp$, and \$ be new symbols. Let $\Gamma_\sharp := \Gamma \cup \{\sharp\}$ and $\overline{\Gamma}_\sharp := \overline{\Gamma} \cup \{\sharp\}$. Define

$$W_\sharp := \overline{\Gamma}_\sharp^* \cdot \Gamma_\sharp^* \cap ((\sharp^{\leqslant 2} \cdot ((\overline{\Gamma} \cup \Gamma) \cdot \sharp\sharp)^* \cdot (\overline{\Gamma} \cup \Gamma) \cdot \sharp^{\leqslant 2}) \cup \sharp^{\leqslant 2}),$$

where $\sharp^{\leqslant 2}$ is a shorthand for $\{\square, \sharp, \sharp\sharp\}$.

As can be seen, this regular language consists of words, where between letters of Γ or $\overline{\Gamma}$ there are always exactly two $\sharp$'s. This language will not only be used to define the compression alphabet, but also to make some definitions during the construction process easier. The purpose of the $\sharp$'s will be explained later.

Definition 4. Let $\mu_l = \max\{|u| \mid u \in \mathrm{dom}(R)\}$ and $\mu_r = \max\{|v| \mid v \in \mathrm{range}(R)\}$ be the maximum length of the respective rule sides. Because R is length-reducing $\mu_l > \mu_r$ holds. Let $\mu := \max\{\mu_l, |t_l|, |t_r|\}$. The *compression alphabet* Γ_1 is defined as:

$$\Gamma_1 := \{\xi_w \mid w \in W_\sharp \wedge 1 \leqslant |w| \leqslant 3\mu + 5\}.$$

The elements of Γ_1 are called *compression letters*. For distinction, letters in the index of a compression letter will be called *index letters*.

Remark 3. At least $\mu + 1$ letters of the original alphabet Γ can be stored in one compression letter.

If we need to delete the surplus $\sharp$'s, we use the following morphism:

Definition 5. Let $\hat{a}$ be the morphism $\hat{a} : (\Gamma_\sharp \cup \overline{\Gamma}_\sharp \cup \{¢, \$\})^* \to (\Gamma \cup \overline{\Gamma} \cup \{¢, \$\})^*$ defined by:

$$\hat{x} := \begin{cases} \square & x = \sharp \\ x & \text{otherwise.} \end{cases}$$

Sometimes it is necessary to extract the information of the subscripts in a word from Γ_1^*:

Definition 6. Let *hata* be the morphism $\hat{a} : (\Gamma_1 \cup \overline{\Gamma} \cup \Gamma \cup \{¢, \$\})^* \to (\overline{\Gamma}_\sharp \cup \Gamma_\sharp)^*$ defined by:

$$\check{x} := \begin{cases} w & x = \xi_w \in \Gamma_1 \\ x & \text{otherwise.} \end{cases}$$

We assume, without loss of generality, that brackets are not in our alphabets so far and use them in the following for better readability.

4.3. Translating the input

The first step in the simulation of C is to translate the input into the compression alphabet. At the same time, we will take care of t_l and t_r. The new end marker letters (not words!) of C' will be $t'_l := ¢$ and $t'_r := \$$.

Short words $w \in L_C, |w| \leqslant 2$ will be handled separately, they are directly added to the language $L_{C'}$. For them, a set R_1 of rules is used:

$$R_1 := \{(¢w\$, Y) \mid w \in L_C \wedge |w| \leqslant 2\}.$$

Obviously, R_1 can be computed easily.

For the translating rule system we will give a new set of rules R_2, but first we have a look at its the mode of operation. Decompose t_l and t_r into single letters in the following way: $t_l = a_1 a_2 \cdots a_{|t_l|}$ and $t_r = c_1 c_2 \cdots c_{|t_r|}$. R_2 will be designed to be a confluent and weight-reducing rewriting system such that for every $w \in \Sigma^{\geqslant 2}$ with $w = b_1 b_2 \cdots b_i \cdots b_{|w|}, b_i \in \Sigma (1 \leqslant i \leqslant |w|)$ we can make the following reduction with R_2:

$$¢w\$ \underset{R_2}{\to^*} ¢\xi_{(\overline{a}_1\sharp\sharp)(\overline{a}_2\sharp\sharp)\cdots(\overline{a}_{|t_l|}\sharp\sharp)(\overline{b}_1\sharp\sharp)}\xi_{b_2\sharp\sharp}\xi_{b_3\sharp\sharp}\cdots\xi_{b_{|w|-1}\sharp\sharp}\xi_{(b_{|w|}\sharp\sharp)(c_1\sharp\sharp)(c_2\sharp\sharp)\cdots(c_{|t_r|}\sharp\sharp)}\$,$$

where we ensure that R_2 does not do more than such translations. Especially, the right-hand sides of the reductions given above are required to be irreducible in R_2. Furthermore, require that the first translated letter after the ¢ always appears in the last reduction step. This is necessary to give a precise moment in the reduction after which the rule sets defined in the following parts of the construction can begin to work.

After this explanation, the following definition of R_2 should be clear. Assume t_l and t_r to be composed of letters a_i and c_i as above. R_2 will be composed of the two parts $R_{2,1}$ and $R_{2,2}$. We start with the translation from the right with the system $R_{2,1}$:

$$R_{2,1} := \{(def\$, de\xi_{(f\sharp\sharp)(c_1\sharp\sharp)(c_2\sharp\sharp)\cdots(c_{|t_r|}\sharp\sharp)}\$) \mid d,e,f \in \Sigma\}$$
$$\cup \{(de\xi_{(f\sharp\sharp)(c_1\sharp\sharp)\cdots(c_{|t_r|}\sharp\sharp)}\$, d\xi_{e\sharp\sharp}\xi_{(f\sharp\sharp)(c_1\sharp\sharp)\cdots(c_{|t_r|}\sharp\sharp)}\$) \mid d,e,f \in \Sigma\}$$
$$\cup \{(de\xi_{f\sharp\sharp}, d\xi_{e\sharp\sharp}\xi_{f\sharp\sharp}) \mid d,e,f \in \Sigma\}.$$

Now, the translation has to be finished:

$$R_{2,2} := \{(\not{c}e\xi_{f\sharp\sharp}, \not{c}\xi_{(\overline{a}_1\sharp\sharp)(\overline{a}_2\sharp\sharp)\cdots(\overline{a}_{|t_l|}\sharp\sharp)(\overline{e}\sharp\sharp)}\xi_{f\sharp\sharp}) \mid e,f \in \Sigma\}.$$

Note that if $t_l = \square$ then the over-lined e is the first index letter after the translation. Finally, $R_2 := R_{2,1} \cup R_{2,2}$.

Example 3. Let $t_l = dd, t_r = c$, and $w = abad$. Then R_2 translates the input to:

$$\not{c}\xi_{\overline{d}\sharp\sharp\overline{d}\sharp\sharp\overline{a}\sharp\sharp}\xi_{b\sharp\sharp}\xi_{a\sharp\sharp}\xi_{d\sharp\sharp c\sharp\sharp}\$.$$

The following can easily be verified:

Claim 1. *$R_1 \cup R_2$ is confluent and, with a suitable weight function, weight-reducing (since each of these rules translates exactly one terminal into a nonterminal, we simply have to assign a weight which is big enough to the terminals). The last rule applied from $R_1 \cup R_2$ is either an accepting rule from R_1 or a rule from $R_{2,2}$. Thus, the first time an over-lined letter appears in the process is after the complete input has been translated into the compression alphabet.*

The rightmost over-lined letter marks the position at which the simulation of C will work with the following rule sets. In general, the rightmost over-lined index letter of a compressed word can be identified with the head position of the automaton described above.

The $\sharp$'s are the *surplus letters* mentioned in the list of construction principles. Moving them to the left or to the right in a suitable manner will be necessary for the weight-reduction property of the rules to follow.

4.4. Simulating shift operations

The next step is similar to the shift operations of automata for CRLs. Sometimes it is necessary to *move right* (that is, shift) the position of a possible next reduction. Note that in order to decide if a shift operation is necessary, an automaton accepting a CRL has to inspect at most μ_l letters of the left-hand stack.

Definition 7. For a word w let $\mathrm{Suff}(w)$ be the set of all suffixes of w.

Definition 8. Given the rewriting system R, we will call a word $w \in I_{\mathrm{RR}}(R)$ a *shift suffix*, if $|w| \leqslant \mu_l$ and w is not a suffix of the left-hand side of a rule of R. Let S_R be the set of all shift suffixes of R.

Whenever the automation accepting the language of C finds a shift suffix on the top of its left-hand stack, it must perform a shift operation, as in this case no left-hand side of a rule will be found on the top of the left-hand stack.

Definition 9. Given $R \subseteq \Gamma^+ \times \Gamma^*$, a *sufficient set of shift suffixes* is a set $S \subseteq S_R$ satisfying:

$$\forall w \in I_{RR}(R) \cap \Gamma^+ : ((\exists u \in \mathrm{dom}(R) : w \in \mathrm{Suff}(u)) \vee (\exists w' \in \Gamma^*, u \in \Gamma^+ : w = w'u \wedge u \in S)).$$

Definition 10. The following set of shift suffixes will be denoted with S:

$$S := \{w \mid w \in S_R \wedge \nexists w' \in \mathrm{Suff}(w) \setminus \{w\} : w' \in S_R\}.$$

Accordingly, $\overline{S}$ is the image of S under $^-$.

The next two propositions should be intuitive.

Proposition 4.4.1. *S is a sufficient set of shift suffixes.*

Proposition 4.4.2. *S is minimal: Any sufficient set of shift-suffixes S' is a superset of S or equal to it.*

Since in our new systems all reductions have to take place *within the indices* and cannot directly work on the letters of the original alphabets we need to take care of this. Especially, any (sub)-word of original letters can be distributed over more than one letter of the compression alphabet, and the $\sharp$'s have to be considered, too. So, we *translate* S in the following way:

Definition 11. Define the S_ξ set of *translated shift suffixes* as:

$$\begin{aligned} S_\xi := \{\xi_{w_1}\xi_{w_2}\cdots\xi_{w_{n-1}w_n} \mid\ & 2 \leqslant n \wedge w_i \in W_\sharp \cap \overline{\Gamma}_\sharp^+ \ (1 \leqslant i \leqslant n-2) \\ & \wedge w_{n-1} \in W_\sharp \cap \overline{\Gamma}_\sharp^* \\ & \wedge w_n \in W_\sharp \cap (\Gamma \cdot \Gamma_\sharp^*) \\ & \wedge w_1 w_2 \cdots w_n \in W_\sharp \\ & \wedge (\exists v_1, v_2 : v_1 v_2 = \hat{w}_1 \wedge v_2 \hat{w}_2 \cdots \hat{w}_{n-1} \in \overline{S}) \\ & \wedge \hat{w}_2 \cdots \hat{w}_{n-1} \notin \overline{S}\}. \end{aligned}$$

It is also possible that the simulated left-hand store does contain an irreducible word which is not a shift suffix. This happens when the word in the store is too short and a suffix of a left-hand-side of a rule. This case is handled with the following set.

Definition 12. The set of *short shift words* is defined as:

$$\begin{aligned} S_\not\in := \{\not\in\xi_{w_1}\xi_{w_2}\cdots\xi_{w_{n-1}w_n} \mid\ & 2 \leqslant n \wedge w_i \in W_\sharp \cap \overline{\Gamma}_\sharp^+ \ (1 \leqslant i \leqslant n-2) \\ & \wedge w_{n-1} \in W_\sharp \cap \overline{\Gamma}_\sharp^* \\ & \wedge w_n \in W_\sharp \cap (\Gamma \cdot \Gamma_\sharp^*) \\ & \wedge w_1 w_2 \cdots w_n \in W_\sharp \\ & \wedge (\exists u \in \mathrm{dom}(R), u' \in \mathrm{Suff}(u) \cap I_{RR}(R) : \hat{w}_1 \hat{w}_2 \cdots \hat{w}_{n-1} = \overline{u'})\}. \end{aligned}$$

In order to get a confluent rewriting system in combination with the rules defined in the later subsections, we have to use a kind of lookahead set. We will discuss its exact role in the following subsections and simply give the definition here:

Definition 13. The *lookahead set* S_l is defined as

$$\begin{aligned} S_l := &\{\xi_{w_1}\xi_{w_2} | w_1 w_2 \in W_\sharp \wedge \hat{w}_1\hat{w}_2 \in \Gamma^*\} \\ &\cup \{\xi_{w_1\sharp\sharp}\$ | w_1\sharp\sharp \in W_\sharp \wedge \hat{w}_1 \in \Gamma^*\} \\ &\cup \{\$\}. \end{aligned}$$

Furthermore, we use the following two abbreviations:

Definition 14.

$$W_{\sharp,\$} := W_\sharp \cup ((W_\sharp \cap W_\sharp \cdot \{\sharp\sharp\}) \cup \{\sharp, \square\}) \cdot \{\$\}.$$

That means, each word in $W_{\sharp,\$}$ is an arbitrary word from $W_\sharp$, or it is a word from $W_\sharp$ ending with two $\sharp$'s concatenated with \$, or it is the word $\sharp\$$, or it is simply a \$.

Definition 15. The set $W_{\sharp,\$,\Gamma}$ is defined as the subset of $W_{\sharp,\$}$ whose words start with a letter from Γ:

$$W_{\sharp,\$,\Gamma} := W_{\sharp,\$} \cap \Gamma \cdot W_\sharp \cdot \{\square, \$\}.$$

Now we construct R_3: For each $w = \xi_{w_1}\xi_{w_2}\cdots\xi_{w_{n-2}}\xi_{w_{n-1}w_n}$ with $w \in S_\xi$ or $\not\in w \in S_\not\in$ and $w' \in S_l$ such that $\check{w}\check{w}' \in W_{\sharp,\$}$ we add a rule $r_{w,w'}$ to the new system R_3. Assume that $w_n = aw'_n$ with $a \in \Gamma$ and $w'_n \in W_\sharp$. If $w \in S_\xi$ the rule to be added for w and w' is

$$r_{w,w'} := (\xi_{w_1}\xi_{w_2}\cdots\xi_{w_{n-2}}\xi_{w_{n-1}w_n}w', \xi_{w_1}\xi_{w_2}\cdots\xi_{w_{n-2}}\xi_{w_{n-1}\bar{a}w'_n}w').$$

If $\not\in w \in S_\not\in$ we add the rule

$$r_{w,w'} := (\not\in\xi_{w_1}\xi_{w_2}\cdots\xi_{w_{n-2}}\xi_{w_{n-1}w_n}w', \not\in\xi_{w_1}\xi_{w_2}\cdots\xi_{w_{n-2}}\xi_{w_{n-1}\bar{a}w'_n}w').$$

Note that the cases $w \in S_\xi$ and $\not\in w \in S_\not\in$ are disjoint.

Remark 4. If a shift is necessary but no next letter without over-lining exists no next reduction is possible. In such a case the simulated system also would reduce to an irreducible word.

Claim 2. *R_3 is confluent.*

Proof. The left-hand sides of rules derived from the translated shift words in S_ξ have no nontrivial overlaps with the rules derived from short shift words in $S_\not\in$. Because S is minimal, left-hand sides of rules derived from S_ξ do not overlap with each other. Left-hand side of rules from $S_\not\in$ do not overlap because of the $\not\in$. So, there are no nontrivial overlaps between left-hand rule sides in R_3. Therefore it is confluent with Theorem 2.1. □

It is also possible to show that R_3 is weight-reducing. The matter of weight-reduction will be discussed later, at the moment simply assume that over-lined letters in the subscripts of compression letters add slightly less to the weight.

Claim 3. *$R_1 \cup R_2 \cup R_3$ is confluent and weight-reducing. There are no overlaps between the former two and R_3, because on the left-hand sides of R_1 and R_2 rules there are never over-lined letters.*

Example 4. Assume $bba \in S$. Then $w = \xi_{\overline{b}\sharp\sharp\overline{b}\sharp\sharp\overline{a}\sharp}\xi_{\sharp c\sharp} \in S_\xi$. Assume $w' = \xi_\sharp\$$. This leads to the rule: $r_w = (\xi_{\overline{b}\sharp\sharp\overline{b}\sharp\sharp\overline{a}\sharp}\xi_{\sharp c\sharp}\xi_\sharp\$, \xi_{\overline{b}\sharp\sharp\overline{b}\sharp\sharp\overline{a}\sharp}\xi_{\sharp\overline{c}\sharp}\xi_\sharp\$) \in R_3$.

4.5. The "weight-spreading" technique

Now we reach the core of the construction, which will be called *weight-spreading*. The main idea is to simulate rules of R piecewise. This simulation is similar to the construction of a context-sensitive grammar from a monotone one [7]. In order to achieve a weight-reducing system a second principle is used: the simulation will reduce the length of the subscripts of the compression letters.

In order to make the construction more understandable, we provide two examples. The first is *not working as desired*, it is used to illustrate why two $\sharp$'s are put between the letters of $\overline{\Gamma}$ or Γ, respectively. The second example shows the correct construction at work.

Example 5. Consider the rule $(aaaa, bbb)$ and assume we had used only a single $\sharp$ as surplus letter. The following possible reduction could be used (each step being identified with a rule) (see Fig. 1).

In the first rewriting step we use the known shift position (the last over-lined subscript letter) to assert that no other rules can interfere. During the next three steps we simulate the actual reduction. Note that the first b also is over-lined. This is possible because after each reduction in the corresponding automaton a shift is necessary, before another reduction can take place.

This example shows the problems we have to deal with. A look at the last letter in the first line and the last letter in the last line reveals that the length of the subscript did not change. In consequence, if we would do this for all rules in any CRLS, this could cause weight-reduction problems.

Example 6. Therefore, we now change the example insofar, as we introduce double $\sharp$'s. Let the rule in question again be $(aaaa, bbb)$ (see Fig. 2).

The $\sharp$'s are used to spread the length-reduction of the original rule over the compression letters in the simulation. Observe that (especially) in the last step of the example the result contains three

$$\xi_{\overline{a}\sharp a}\xi_{\sharp\overline{a}\sharp}\xi_{\overline{a}\sharp a}$$
$$\longrightarrow \qquad \text{1. "lock" with new nonterminal}$$
$$\xi_{\overline{a}\sharp a}\xi_{\sharp\overline{a}\sharp}\xi_t$$
$$\longrightarrow \qquad \text{2. change first letter}$$
$$\xi_{\overline{b}\sharp}\xi_{\sharp\overline{a}\sharp}\xi_t$$
$$\longrightarrow \qquad \text{3. change middle letter}$$
$$\xi_{\overline{b}\sharp}\xi_{b\sharp}\xi_t$$
$$\longrightarrow \qquad \text{4. change last letter, remove lock}$$
$$\xi_{\overline{b}\sharp}\xi_{b\sharp}\xi_{b\sharp a}$$

Fig. 1. An incorrect simulation.

$$\xi_{\overline{a}\sharp\sharp\overline{a}}\xi_{\sharp\sharp\overline{a}\sharp\sharp}\xi_{\overline{a}\sharp\sharp a}$$
$$\longrightarrow \qquad \text{1. “lock” with new nonterminal}$$
$$\xi_{\overline{a}\sharp\sharp\overline{a}}\xi_{\sharp\sharp\overline{a}\sharp\sharp}\xi_{t}$$
$$\longrightarrow \qquad \text{2. change first letter}$$
$$\xi_{\overline{b}\sharp\sharp}\xi_{\sharp\sharp\overline{a}\sharp\sharp}\xi_{t}$$
$$\longrightarrow \qquad \text{3. change middle letter}$$
$$\xi_{\overline{b}\sharp\sharp}\xi_{b\sharp\sharp b}\xi_{t}$$
$$\longrightarrow \qquad \text{4. change last letter, remove lock}$$
$$\xi_{\overline{b}\sharp\sharp}\xi_{b\sharp\sharp b}\xi_{\sharp\sharp a}$$

Fig. 2. A correct simulation.

compression letters. More than three *active letters* are not necessary during any rule simulation, because a right-hand side of a rule and the added $\sharp$'s fit into one compression letter (to be exact, sometimes some *unchanged context* appears on the right-hand sides of the simulation rules for the sake of confluence, necessitating four letters). Basically, the weight of a compression letter will be computed from the number of its index letters (over-lined letters add *slightly* less to the weight). Therefore in the worst case – when the original rule has a length-reduction of one – we reduce the number of index letters by three and spread this reduction over the resulting three compression letters. This is the cause for adding two $\sharp$'s after each index letter of $\Gamma \cup \overline{\Gamma}$ during the translation with R_2.

For generalizing this example, a lot of cases have to be handled. The main idea is to identify all possibilities how the left-hand side of an original rule can be split over one or more letters of the compression alphabet. It should be noted that sometimes some index letters (from $\overline{\Gamma}$) of the first compression letter do not change, which corresponds to unchanged parts of the simulated left-hand stack. Furthermore, the question of finding the right place for the reduction to work has to be handled.

At this point we can explain the necessity of the lookahead set for our construction. Assume there is another rule whose left-hand side ends with a letter b. As soon as the middle letter is changed in our example (step 3), a simulation of that second rule could start, therefore possibly preventing the ξ_t from being removed for the rest of the simulation. This would result in a rewriting system which is not confluent. So our example only showed the simulation part of our construction, but not how the confluence is achieved. In the simulation rules which we are going to add, this is prevented by looking two compression letters further whether there is a locking nonterminal. If there are no two compression letters, but only one and the end marker letter \$, or even only the \$, the same applies. How this works can be seen in detail in the following.

4.6. Case distinctions

Now, we will give the necessary distinction of main cases. The sub-cases of those are given in Appendix A.

4.6.1. Simulation in one letter

In some cases, the simulation is very easy. Especially, when the complete reduction can be simulated within one letter of the compression alphabet. We handle this first case with the following set:

$$\begin{aligned} T_1 := \{(u, v, \xi_{w_1w_2w_3}w) \mid\ & (u, v) \in R \\ & \wedge w_1w_2w_3 \in W_\sharp \\ & \wedge w \in S_I \\ & \wedge w_1 \in \overline{\Gamma}_\sharp^* \\ & \wedge \hat{w}_2 = \overline{u} \wedge w_2 \in \overline{\Gamma} \cdot \overline{\Gamma}_\sharp^* \cdot \overline{\Gamma} \cdot \{\sharp\sharp\} \\ & \wedge w_3\check{w} \in W_{\sharp,\$,\Gamma}\}. \end{aligned}$$

Now, for each element $t \in T_1$ assume $u = a_1 \cdots a_{|u|}, a_i \in \Gamma\ (1 \leqslant i \leqslant |u|)$ and $v = b_1b_2 \cdots b_{|v|}, b_i \in \Gamma\ (1 \leqslant i \leqslant |v|)$. For each t add a rule r_t to a new system R_4:

$$r_t := (\xi_{w_1w_2w_3}w, \xi_{w_1(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_3}w).$$

If $w_1 = w_3 = v = \square$, this would produce a letter $\xi_\square$, which is not in the compression alphabet. In these cases identify $\xi_\square \equiv \square$. Then the rule will be simply deleting one symbol.

4.6.2. Simulation of complex cases

Now the difficult part of the construction will be discussed. What has to be done, if the left-hand side of the original rule is not in one letter but distributed over the indices of several compression letters? Again we use a set which contains all cases of possible rule applications that are not covered by the above set T_1:

$$\begin{aligned} T_2 := \{(u, v, \xi_{w_1w_2}\xi_{w_3}\xi_{w_4} & \cdots \xi_{w_{n-2}}\xi_{w_{n-1}w_n}w) \mid (u, v) \in R \\ & \wedge 4 \leqslant n \\ & \wedge w_1 \cdots w_n \in W_\sharp \\ & \wedge w \in S_I \\ & \wedge w_1 \in \overline{\Gamma}_\sharp^* \\ & \wedge w_2w_3 \cdots w_{n-1} \in \overline{\Gamma} \cdot \overline{\Gamma}_\sharp^* \cdot \overline{\Gamma} \cdot \{\sharp\sharp\} \\ & \wedge w_2 \neq \square \\ & \wedge \hat{w}_2\hat{w}_3 \cdots \hat{w}_{n-1} = \overline{u} \\ & \wedge w_n\check{w} \in W_{\sharp,\$,\Gamma}\}. \end{aligned}$$

For each $t \in T_2$ (T_2 is finite) a set of rules is added to R_4.

This also needs some further nonterminals. These will be collected in the set Γ_2, whose elements will be called *locking symbols*. Again, we assume $u = a_1 \cdots a_{|u|}, a_i \in \Gamma\ (1 \leqslant i \leqslant |u|)$ and $v = b_1b_2 \cdots b_{|v|}, b_i \in \Gamma\ (1 \leqslant i \leqslant |v|)$. Whenever we speak of three or four rules these are a simulation of an original rule in as many steps.

The following 10 cases have to be dealt with, some of them with further sub-cases. Details can be found in Appendix A.

Let $t = (u, v, \xi_{w_1w_2}\xi_{w_3}\xi_{w_4} \cdots \xi_{w_{n-2}}\xi_{w_{n-1}w_n}w) \in T_2$:

1. $n = 4, w_3 = \square$ (already covered by the rules for T_1),
2. $n = 4, v = \square, w_3 \neq \square$ (all deleting rules need some extra care, four sub-cases),

3. $n = 4, v \neq \square, w_3 \in \{\sharp, \sharp\sharp\}$ (similar to the rules for T_1, no sub-cases, one rule for t),
4. $n = 4, v \neq \square, |w_3| > 2$ (five sub-cases, up to three rules for t),
5. $n = 5, v \neq \square, w_4 = \square$ (already captured by 3. and 4.),
6. $n = 5, v = \square, w_4 \neq \square$ (again deleting rule, four sub-cases),
7. $n = 5, v \neq \square, w_4 \in \{\sharp, \sharp\sharp\}$ (seven sub-cases, up to three rules for t),
8. $n = 5, v \neq \square, w_3 \in \{\sharp, \sharp\sharp\}, |w_4| > 2$ (three sub-cases, up to three rules for t),
9. $n = 5, v \neq \square, |w_3| > 2, |w_4| > 2$ (23 sub-cases, up to four rules for t),
10. $n > 5$. (The rule itself is not simulated, instead the substring $\xi_{w_3} \cdots \xi_{w_{n-2}}$ is compressed into one letter $\xi_{w_3 \cdots w_{n-2}}$; observe that $|\xi_{w_3} \cdots \xi_{w_{n-2}}| < \mu$. Thus, the cases $n > 5$ are reduced to the cases $n \leqslant 5$.)

4.7. Final rules

Since the set of final rules is very simple, we already describe it here:

$$R_5 := \left\{ (\text{¢}w\$, Y) | w \in \Gamma_1^{\leqslant 3}, \check{w} = \overline{Y}\sharp\sharp \right\}.$$

4.8. An example case

In order to show how the sub-cases of the main cases above can be derived, one example is provided. This example case is Case 9p in Appendix A.

Example 7. Consider $t = (u, v, \xi_{w_1 w_2} \xi_{w_3} \xi_{w_4 w_5} w) \in T_2$, so $n = 5$. Assume $|w_3| > 2$ and $|w_4| > 2$. We know $\hat{w}_2 \hat{w}_3 \hat{w}_4 = \overline{u}$. Furthermore, decompose u and v into letters: $u = a_1 \cdots a_{|u|}, v = b_1 \cdots b_{|v|}$.

Then we know there exist $i, j > 0$ such that $j > i$ and $\hat{w}_2 = \overline{a}_1 \cdots \overline{a}_i$, $\hat{w}_3 = \overline{a}_{i+1} \cdots \overline{a}_j$, and $\hat{w}_4 = \overline{a}_{j+1} \cdots \overline{a}_{|u|}$.

We use i and j to determine how letters have to be spread over the subscripts of the new compression letters. First, we compute indices k and l which would simply assign the letters of v from right to left and from the third to the first compression letter. Let $k := |v| - |u| + i$ and $l := |v| - |u| + j$. The variables k and l can be conveniently used in the construction of the rules, as can be seen in Appendix A. We also use them in the case distinction itself to emphasize their importance.

Obviously, $l > k$ always holds. In Fig. 3 the relation between i, j, k, and l is illustrated. As one can see, l and k allow to determine over how many compression letters the reduction result can be spread with respect to w_2, w_3, and w_4.

$\hat{w}_2 = \overline{a}_1 \cdots \overline{a}_i$	$\hat{w}_3 = \overline{a}_{i+1} \cdots \overline{a}_j$	$\hat{w}_4 = \overline{a}_{j+1} \cdots \overline{a}_{\|u\|}$	
$\overline{b}_1 b_2 \cdots b_i$	$b_{i+1} \cdots b_j$	$b_{j+1} \cdots b_{\|v\|}$	$l > k > 0$ (3 comp. l.)
$\overline{b}_1 b_2 \cdots b_i$	$b_{i+1} \cdots b_{\|v\|}$		$l > 0, k \leq 0$ (2 compression letters)
$\overline{b}_1 b_2 \cdots b_{\|v\|}$			$l, k \leq 0$ (1 compression letter)

Fig. 3. Identifying sub-cases for rule simulation with i, j, k, and l.

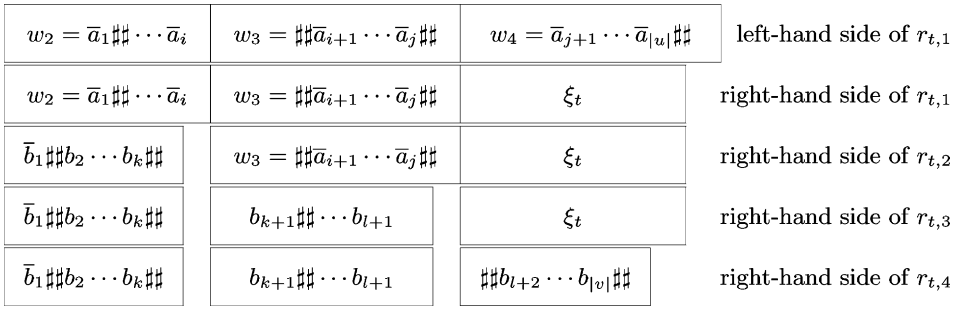

Fig. 4. Distribution of letters in the simulation.

Additionally, we have to consider the $\sharp$'s at the split points between w_2 and w_3, respectively between w_3 and w_4. It is clear that there exist w_2', w_2'', w_3', w_3'', w_3''',w_4', and w_4'' such that $w_2 = w_2'w_2'', w_3 = w_3'w_3''w_3''', w_4 = w_4'w_4''$, with $w_2''w_3' = \sharp\sharp$ and $w_3'''w_4' = \sharp\sharp$.

Now the sub-cases depend on $k > 0$ or not, $l > 0$ or not, the length of w_2'', and the length of w_3'''. Example 6 above leads to $k = 1$, $l = 2$, $w_2'' = \square$, and $w_3''' = \sharp\sharp$. In this sub-case we introduce a new nonterminal ξ_t (for each t a separate one), which is added to Γ_2, and the following four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_{\sharp\sharp(b_{l+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

Fig. 4 illustrates the distribution of letters in the simulation, the length of the boxes is indicating their weight. Of the surplus letters $\sharp$ only the most important ones are explicitly shown.

Observe that as soon as the ξ_t is introduced, the distance between the last compression letter containing a letter from $\overline{\Gamma}$ and the ξ_t is less than two. Therefore, until the ξ_t is removed again, no further simulation of a rule can be started. In the following section we discuss the weight function and other correctness issues.

5. Correctness of the construction

5.1. Weight-reduction

In order to show that R_4 is weight-reducing, we need a suitable weight function. The idea is to distribute the weights of the right-hand rule sides v in the original system R over two or more compression letters. Therefore, the strategy for the construction is called "weight-spreading." The most important part of a weight function ψ are the weights defined for letters from Γ_1. The weights for Σ, $\{\not\in, \$\}$, and for Γ_2 can be easily found based on this. Let $\varphi(x) : W_\sharp \to \mathbf{N}$ be the weight function defined by:

$$\varphi(x) := \begin{cases} 2\mu_l + 2 & (x \in \Gamma) \\ 2\mu_l & \text{otherwise.} \end{cases}$$

Then $\psi(x) : \Gamma_1 \to \mathbf{N}$ is defined by $\psi(\xi_w) := \varphi(w) + 1$. The following property can be verified easily:

Claim 4. *For all* $\xi_v, \xi_w \in \Gamma_1$*:* $|v| > |w| \Rightarrow \psi(\xi_v) > \psi(\xi_w)$ *and* $\xi_{vw} \in \Gamma_1 \Rightarrow \psi(\xi_v) + \psi(\xi_w) > \psi(\xi_{vw})$.

So, ψ is the Γ_1 part of the required weight function. For Γ_2 the fact can be used that all Γ_1 weights are odd, so blocking letters from Γ_2 will have even weight just fitting "in between."

By giving $\not\in, \$$, and Y the weight 1 and by assigning the weight $1 + \max\{\psi(x) | x \in \Gamma_1\}$ to the terminals (Σ) we get a weight function ψ which is appropriate for all our rules.

5.2. Confluence of the simulation

Confluence of R_4 is ensured by three properties:

1. The place of any possible next reduction is uniquely given by the last overlined index letter.
2. We assumed without restriction of generality that the original rewriting system R is normalized. Therefore, for all $t, t' \in T_1 \cup T_2, t = (u, v, w), t' = (u', v', w')$ we know $w = w' \Rightarrow u = u' \wedge v = v' \Rightarrow t = t'$.This implies that whenever the overlined index letters form a reducible word, exactly one t can be used.
3. After introducing the ξ_t, the only possible reduction steps are given by the simulation of the original rule. This is ensured by the lookahead: ξ_t cannot be further right from the compression letter containing the last overlined letter than two compression letters. Therefore as soon as ξ_t is introduced, no other simulation can start before it is removed again. Also, no single step of the simulation can be applied twice, and the order of applying the simulation rules is fixed. This holds because the structure of the index words w.r.t. the $\sharp$'s prevents permutations of the rules.
4. The rules for Case 10. ($n > 5$) cannot interfere with each other and also, because of the lookahed, not with the simulation rules.

So, R_4 itself is confluent (there can be no critical pairs, compare to [4] or [11]).

Remark 5. As we will see later, assuming a normalized rewriting system is not only important for confluence but also for equivalence of the newly constructed csCRLS to the original CRLS.

5.3. Overall correctness

Lemma 1. *Let* $\Gamma' := \Sigma \cup \{\not\in, \$, Y\} \cup \Gamma_1 \cup \Gamma_2$, $\Sigma' := \Sigma$, $R' := R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5$. *Then* $C' := (\Gamma', \Sigma', R', \not\in, \$, Y)$ *is a csCRLS and* $L_{C'} = L_C$.

Proof. There are five steps necessary: Is C' well defined? Is it weight-reducing? Is it confluent? Does it define the correct language? Is it context-splittable? Again, we refer the reader to Appendix A of this article for the overview of all cases.

1. To check if C' is well defined we only have to assert that nowhere a $\xi_\square$ would be necessary in R_4. By observing the relations between the subscript word lengths and n, i, j, k, and l of the subcases mentioned above this can be shown to be true.

2. Checking the weight-reduction of the rules is a rather tedious effort, but straightforward. In Appendix A all rules of R_4 are described.
3. First, observe that the first three parts together (that is, $R_1 \cup R_2 \cup R_3$) are confluent. Second, reduction rules of R_4 cannot overlap with those rules or with rules from R_5 (the latter because $Y \in I_{RR}(R)$). We already showed that R_4 is confluent, so finally R' is confluent.
4. All our rules in R_4 simulate the original system R. Therefore, our new system R' cannot accept words which are not in L_C. Because R is assumed to be normalized, the exact behaviour of an automaton accepting L_C is simulated. Therefore, for each accepting computation of such an automaton, a reduction exists which simulates it. Therefore, every word in L_C is in $L_{C'}$, too, and $L_C = L_{C'}$ holds.
5. Checking the context-splittability is the easiest part, it can be verified by simply looking at all rules.

With this lemma, the proof of the normal form theorem is complete. □

Corollary 5.1. *For each CRL L there is a weight-increasing context-sensitive grammar G such that* $L(G) = L$ *and each rule is context-sensitive in the sense of* [7].

We do not elaborate the proof in full detail, since in [15] a direct grammar construction for the whole class of GCSL, which includes CRL [14], is given.

Proof. Let $C = (\Gamma, \Sigma, R, t_l, t_r, Y)$ be a csCRLS defining L. Basically, one has to swap the rules sides of the rewriting rules of C. There are only two technical problems.

1. Instead of the end-markers one has to mark the left-hand and right-hand ends of the sentential form and adapt all rules using those end-markers accordingly. Of course, this will lead to some more nonterminals.
2. After swapping the sides, there might be rules of the form $\square \to v$. Because any sentential form produced by the grammar contains at least one letter, we simply use all rules of the form $x \to xv$ and $y \to vy$ where x, y are letters from the alphabets of G (but x is not a letter marking the right-hand side of the sentential form and y is not a letter marking the left-hand side).

With this sketch, the construction of G should be clear. □

6. Consequences of the result

6.1. Refinement of linear time bound

One consequence is that the information flow during reductions is underlying stronger restrictions in a csCRLS. Any movement of a letter in either direction needs at least as many rule applications as the distance to be accomplished. Although this is only a refinement of the linear time bound for the reductions in CRLSs it might be handy for proofs.

6.2. Conflict with normalization

One can see that the piecewise simulation of rules leads to a rewriting system that has reducible right-hand sides, in contrast to the normalized rewriting systems.

The context-splittable normal form and the property of being normalized seem to be dual to each other: We conjecture that there is a CRL that does not have a normalized csCRLS. To justify this conjecture, we give the following example. Consider a rule $(abbb, bba)$. The author knows of no way to simulate such a rule by *a single context-splittable* rule. But a simulation using more rules surely would not be normalized, because the simulation steps would have to be linked to each other.

Besides, a length-reducing simulation in more than one step with a normalized system seems to incorporate conflicting goals. [3]

6.3. Length-reduction and context-splittability

Additionally, the construction of a csCRLS heavily uses weight-reduction. This makes the existence of a length-reducing context-splittable normal form doubtful. Anyhow, we do not conjecture that there is a CRL that has no length-reducing csCRLS nor the opposite. Already our characterization result is against intuition, so there might be an even more complicated construction which shows that a length-reducing csCRLS exists for each CRL.

6.4. Relation to GCSL

Some more interesting questions arise because CRL and DGCSL are the same language class, characterized by sDTPDAs. Therefore, the following results can be obtained. All details are described in [15].

By dropping the condition of confluence we obtain the class of the *growing context-sensitive languages* (GCSL, defined in [8]) from the class CRL. A normal form corresponding to the one established here for CRL also holds for GCSL (thus justifying the use of the term *context-sensitive*). This implies that the class of the *acyclic context-sensitive languages* (ACSL) coincides with GCSL (this problem was posed in [5]).

ACSL are those languages which can be described by a context-sensitive grammar whose context-free kernel (the context-free grammar gained by stripping the context from the context-sensitive rules) is acyclic.

Theorem 6.1 [15]. *For each growing context-sensitive grammar G with language L_G there exists an acyclic context-sensitive grammar G' with language $L_{G'} = L_G$.*

6.5. Normal forms for automata accepting CRL

The definition of TPDAs allows a program $\delta : (Q \times \Gamma \times \Gamma) \to (Q \times \Gamma^* \times \Gamma^*)$ (Q are the states, Γ is the work alphabet). By a construction similar to the one for GCSL it should be possible to restrict the possible rule types – even for shrinking or bounded (D)TDPAs – to the following forms:

[3] Originally, Friedrich Otto raised the question of normalized csCRLSs during Theorietag 2001 of the German Gesellschaft für Informatik.

1. $(q,A,B) \to (q',A,C)$,
2. $(q,A,B) \to (q',AC,\square)$,
3. $(q,A,B) \to (q',C,B)$,
4. $(q,A,B) \to (q',\square,CB)$,
5. $(q,A,B) \to (q',C,\square)$,
6. $(q,A,B) \to (q',\square,C)$, or
7. $(q,A,B) \to (q',\square,\square)$,

with $q,q' \in Q, A,B,C \in \Gamma$ and without limiting the expressive power of the respective automata classes (s(D)TPDA,b(D)TPDA). This is a line of further research.

6.6. Derivation graphs

Buntrock [5] discusses *derivation graphs* for words of a growing context-sensitive grammar. The same can be done for CRLSs. For csCRLSs, one can also introduce the notion of derivation trees (under some restrictions), similar to those of context free grammars (for example, see [9]). The advantage of trees over more general graphs is that they are widely used in compiler construction. By showing that derivation trees are possible for words of CRLs, this language class becomes also interesting for practical applications. This could be a line of further research.

We give an example:

Example 8. Let $C = (\{¢,\$,Y,a,b,c\},\{a,b,c\},R,¢,\$,Y)$ be a CRLS with a rewriting system

$$R = \{(abbba,acca),(¢acca\$,Y)\}.$$

Obviously, $L_C = \{abbba,acca\}$.

For the word *abbba* a derivation graph is given in Fig. 5.

Note that we have Y at the top of the graph in order to make it similar to the derivation graphs for grammars. Nevertheless, the arrows point into the direction of the rewriting rules.

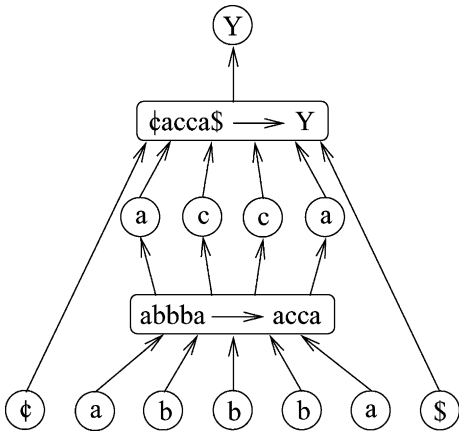

Fig. 5. An example for a derivation graph.

As one can see, the unchanged context of a rewriting step is copied in this graph.

Now we examine a csCRLS C', which differs from C only in the rewriting rules, which are:

$$R = \{(abbba, aca), (¢aca\$, Y)\}.$$

The derivation graph for the word *abbba* still is not a tree, see Fig. 6. We can remove the copied contexts, and get the tree in Fig. 7. This can be understood as a derivation tree which is enriched by rule information. By dropping the nodes containing the rules we get a derivation tree which only differs from a tree for a context-free grammar in the direction of the arrows, see Fig. 8.

It is also possible to remove the nodes ¢ and \$: Except for the last step they always are part of the context.

It is important that we cannot do these steps of removing the contexts if we have deleting rules like $(ccc, \square)$ or $(¢ \cdot c \cdot \$, ¢ \cdot \square \cdot \$)$ (the latter is a deleting rule because in context-splittings ¢ and \$ always have to be part of the contexts). One could work around this problem by introducing

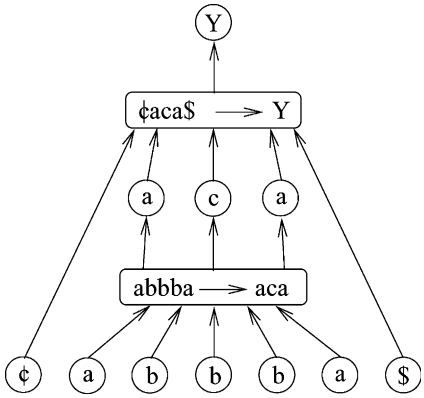

Fig. 6. A derivation graph of a word accepted by a csCRLS.

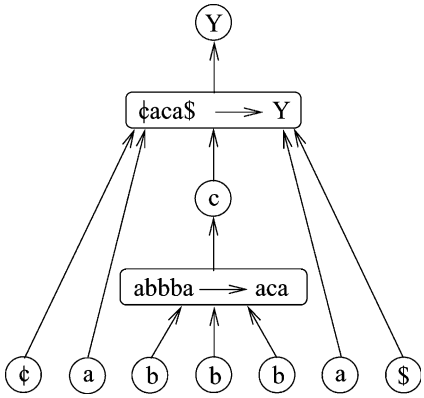

Fig. 7. An enriched derivation tree.

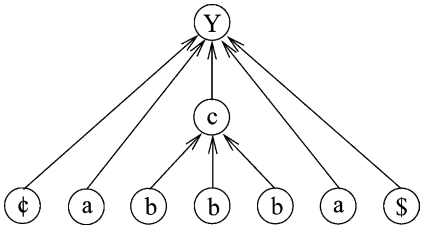

Fig. 8. An example for a derivation tree.

nodes labeled with $\square$. But, either those nodes would be dead ends in the derivation, destroying the tree property, or one would have to insert edges which do not directly correspond to rule applications. For example, simply replace the middle "c" node in Fig. 8, assuming a rule $bbb \to \square$ and a rule $¢aa\$ \to Y$.

7. Conclusion

In this article we have proved that Church–Rosser languages can be described by CRLSs whose rewriting systems very much look like the production sets of context-sensitive grammars.

This result *characterizes deterministic growing context-sensitive languages* and can be transferred to the more general case of GCSL, which is proved in [15]. Basically, the method of weight-spreading leads to a very similar construction for growing context-sensitive grammars. In fact, the generalization to GCSL is even simpler than the deterministic case.

Yet, this alone would not justify to call the result a characterization theorem. But by proving that GCSLs can be described by weight-increasing context-sensitive grammars, one can conclude that ACSL and GCSL coincide. This is a rather surprising result: Although it relatively easy to show that ACSL $\subseteq$ GCSL (see also [5]), the reverse was not expected. All in all, this shows the relevance of our normal form theorem as a characterization of DGCSL.

Acknowledgments

The author wishes to thank the anonymous referees for their helpful remarks.

Appendix A. All cases for the construction of R_4

For each $t \in T_2$ we will give a set of rules that has to be added to R_4. This also uses some further nonterminals. These will be collected in the set Γ_2. Again, we assume $u = a_1 \cdots a_{|u|}$, $a_i \in \Gamma$ $(1 \leqslant i \leqslant |u|)$ and $v = b_1 b_2 \cdots b_{|v|}, b_i \in \Gamma$ $(1 \leqslant i \leqslant |v|)$. Note that $w_2 \neq \square$ is required by the definition of T_2.

There are 10 cases which have to be handled differently:

1. The case $n = 4$ and $w_3 = \square$ is already captured by T_1: Assume $t \in T_2$ and $t = (u, v, \xi_{w_1w_2}\xi_{w_3w_4}w) = (u, v, \xi_{w_1w_2}\xi_{w_4}w)$. Then there is a $w' \in \mathrm{Suff}(\xi_{w_4}w)$ such that $t' = (u, v, \xi_{w_1w_2}w') \in T_1$. Therefore, we do not need to add rules for this case.
2. For $n = 4$, $v = \square$, $w_3 \neq \square$ we get sub-cases:
   (a) $w_1 = \square$, $w_4 = \square$, add one rule which simply deletes two letters:

   $$r_t := (\xi_{w_1w_2}\xi_{w_3w_4}w, w).$$

   (b) $w_1 \neq \square$, $w_4 = \square$ uses one rule:

   $$r_t := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_1}w).$$

   (c) $w_1 = \square$, $w_4 \neq \square$ uses only one rule:

   $$r_t := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_4}w).$$

   (d) $w_1 \neq \square$, $w_4 \neq \square$ uses a new nonterminal xi_t with

   $$\psi(\xi_t) = \psi(\xi_{w_3w_4}) - 1$$

   and three rules:

   $$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_1w_2}\xi_t w)$$
   $$r_{t,2} := (\xi_{w_1w_2}\xi_t w, \xi_{w_1}\xi_t w)$$
   $$r_{t,3} := (\xi_{w_1}\xi_t w, \xi_{w_1}\xi_{w_4}w).$$

3. If $n = 4$, $v \neq \square$, and $w_3 \in \{\sharp, \sharp\sharp\}$, all the action takes place in the first letter. For w_2 there exists a splitting such that $w_2 = w_2'w_2''$ with $w_2''w_3 = \sharp\sharp$. Add the rule:

   $$r_t := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdot(\sharp\sharp b_{|v|})w_2''}\xi_{w_3w_4}w).$$

   This case uses no new nonterminals.
4. If $n = 4$, $v \neq \square$, and $|w_3| > 2$ (then w_3 contains at least one letter from $\overline{\Gamma}$), the reduction must be split over two nonterminals. For w_2 and w_3 there exist splittings such that $w_2 = w_2'w_2''$ and $w_3 = w_3'w_3''$ with $w_2''w_3' = \sharp\sharp$. Since both w_2 and w_3 are nonempty, we know that there exists an $i, 1 \leqslant i < |u|$ with $w_2' = \overline{a}_1\sharp\sharp\cdots\overline{a}_i$ and $w_3'' = \overline{a}_{i+1}\sharp\sharp\cdots\overline{a}_{|u|}\sharp\sharp$. Now there are five sub-cases, depending on the length of w_1, w_2'', and i:Let $k := |v| - |u| + i$. If $k > 0$ this will be used to calculate a split point for the compressed word which is to be substituted. In some cases we use a new nonterminal ξ_t, which will be added to Γ_2. Let the weight of ξ_t be $\psi(\xi_t) := \psi(\xi_{w_3w_4}) - 1$.
   (a) If $k \leqslant 0$ and $w_1 = \square$ add the rule

   $$r_t := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_4}w).$$

   (b) If $k \leqslant 0, w_2'' = \square$, and $w_1 \neq \square$ add the following three rules:

   $$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_1w_2}\xi_t w)$$
   $$r_{t,2} := (\xi_{w_1w_2}\xi_t w, \xi_{w_1}\xi_t w)$$
   $$r_{t,3} := (\xi_{w_1}\xi_t w, \xi_{w_1}\xi_{(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_4}w).$$

   (c) If $k \leqslant 0$, $w_2'' \neq \square$, and $w_1 \neq \square$ add the following three rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_1w_2}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_t w, \xi_{w_1\overline{b}_1}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1}\xi_t w, \xi_{w_1\overline{b}_1\sharp\sharp(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_4}w).$$

(d) If $k > 0$ and $w_2'' = \square$ add three rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_1w_2}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_t w, \xi_{w_1(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_k\sharp\sharp)}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_k\sharp\sharp)}\xi_t w, \xi_{w_1(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_k\sharp\sharp)}\xi_{(b_{k+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_4}w).$$

(e) If $k > 0$ and $w_2'' \neq \square$ add:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3w_4}w, \xi_{w_1w_2}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{|v|})\sharp\sharp w_4}w).$$

Observe that the lengths of the subscripts guarantee the weight-reduction obtained by the rules $r_{t,2}$ and $r_{t,3}$.

5. The case $n = 5$, $v \neq \square$, and $w_4 = \square$ is already captured by the cases above so we do not build new rules in this case.
6. If $n = 5$ and $v = \square$ and $w_4 \neq \square$ we get four sub-cases:
   (a) $w_1 = \square$, $w_5 = \square$, add one rule which simply deletes three letters:
   $$r_t := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, w).$$
   (b) $w_1 \neq \square$, $w_5 = \square$ uses one rule:
   $$r_t := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1}w).$$
   (c) $w_1 = \square$, $w_5 \neq \square$ uses only one rule:
   $$r_t := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_5}w).$$
   (d) $w_1 \neq \square$, $w_5 \neq \square$ uses a new nonterminal ξ_t with
   $$\psi(\xi_t) = \psi(\xi_{w_4w_5}) - 1$$
   and three rules:
   $$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
   $$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{w_3}\xi_t w)$$
   $$r_{t,3} := (\xi_{w_1}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{w_5}w).$$
7. For $n = 5$, $v \neq \square$, and $w_4 \in \{\sharp, \sharp\sharp\}$ the complete reduction takes place in the first two nonterminals. This is similar to $n = 4$, $v \neq \square$. These are the sub-cases:
   (a) $w_3 = \sharp$. Then we can make a one rule reduction without introducing a new nonterminal. Add the rule:
   $$(\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{|v|})w_3}\xi_{w_4w_5}w).$$

(b) $w_3 \neq \sharp$. Then w_3 contains at least one letter from $\overline{\Gamma}$, since $w_3 = \sharp\sharp$ would imply $w_4 \notin \{\sharp, \sharp\sharp\}$.For w_2, w_3 and w_4 there exist splittings such that $w_2 = w_2' w_2''$, $w_3 = w_3' w_3'' w_3'''$ and $w_4 = w_4' w_4''$ with $w_2'' w_3' = \sharp\sharp$ and $w_3''' w_4' = \sharp\sharp$. Since both w_2 and w_3 are nonempty, we know that there exists an $i, 1 \leqslant i < |u|$ with $w_2' = \overline{a}_1 \sharp\sharp \cdots \overline{a}_i$ and $w_3'' = \overline{a}_{i+1} \sharp\sharp \cdots \overline{a}_{|u|}$. Now there are five sub-cases, depending on the length of w_1, w_2'', and i:
Let $k := |v| - |u| + i$. If $k > 0$ this will be used to calculate a split point for the compressed word which is to be substituted. In some cases we use a new nonterminal ξ_t, which will be added to Γ_2. Let the weight of ξ_t be $\psi(\xi_t) := \psi(\xi_{w_3}) - 1$.

(i) If $k \leqslant 0$ and $w_1 = \square$ add the rule

$$r_t := (\xi_{w_1 w_2} \xi_{w_3} \xi_{w_4 w_5} w, \xi_{\overline{b}_1 (\sharp\sharp b_2) \cdots (\sharp\sharp b_{|v|}) w_3'''} \xi_{w_4 w_5} w).$$

(ii) If $k \leqslant 0$, $w_2'' = \square$, and $w_1 \neq \square$ add the following three rules:

$$r_{t,1} := (\xi_{w_1 w_2} \xi_{w_3} \xi_{w_4 w_5} w, \xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,2} := (\xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w, \xi_{w_1} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,3} := (\xi_{w_1} \xi_t \xi_{w_4 w_5} w, \xi_{w_1} \xi_{\overline{b}_1 (\sharp\sharp b_2) \cdots (\sharp\sharp b_{|v|}) w_3'''} \xi_{w_4 w_5} w).$$

(iii) If $k \leqslant 0$, $w_2'' \neq \square$, and $w_1 \neq \square$ add the following three rules:

$$r_{t,1} := (\xi_{w_1 w_2} \xi_{w_3} \xi_{w_4 w_5} w, \xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,2} := (\xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w, \xi_{w_1 \overline{b}_1} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,3} := (\xi_{w_1 \overline{b}_1} \xi_t \xi_{w_4 w_5} w, \xi_{w_1 \overline{b}_1} \xi_{(\sharp\sharp b_2) \cdots (\sharp\sharp b_{|v|}) w_3'''} \xi_{w_4 w_5} w).$$

(iv) If $k > 0$ and $w_2'' = \square$ add three rules:

$$r_{t,1} := (\xi_{w_1 w_2} \xi_{w_3} \xi_{w_4 w_5} w, \xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,2} := (\xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w, \xi_{w_1 (\overline{b}_1 \sharp\sharp)(b_2 \sharp\sharp) \cdots (b_k \sharp\sharp)} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,3} := (\xi_{w_1 (\overline{b}_1 \sharp\sharp)(b_2 \sharp\sharp) \cdots (b_k \sharp\sharp)} \xi_t \xi_{w_4 w_5} w, \xi_{w_1 (\overline{b}_1 \sharp\sharp)(b_2 \sharp\sharp) \cdots (b_k \sharp\sharp)} \xi_{b_{k+1} (\sharp\sharp b_{k+2}) \cdots (\sharp\sharp b_{|v|}) w_3'''} \xi_{w_4 w_5} w).$$

(v) If $k > 0$ and $w_2'' \neq \square$ add:

$$r_{t,1} := (\xi_{w_1 w_2} \xi_{w_3} \xi_{w_4 w_5} w, \xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,2} := (\xi_{w_1 w_2} \xi_t \xi_{w_4 w_5} w, \xi_{w_1 \overline{b}_1 (\sharp\sharp b_2) \cdots (\sharp\sharp b_{k+1})} \xi_t \xi_{w_4 w_5} w)$$
$$r_{t,3} := (\xi_{w_1 \overline{b}_1 (\sharp\sharp b_2) \cdots (\sharp\sharp b_{k+1})} \xi_t \xi_{w_4 w_5} w, \xi_{w_1 \overline{b}_1 (\sharp\sharp b_2) \cdots (\sharp\sharp b_{k+1})} \xi_{(\sharp\sharp b_{k+2}) \cdots (\sharp\sharp b_{|v|}) w_3'''} \xi_{w_4 w_5} w).$$

8. For $n = 5$, $v \neq \square$, $w_3 \in \{\sharp, \sharp\sharp\}$, and $w_4 \notin \{\square, \sharp, \sharp\sharp\}$ there are less sub-cases. Then w_4 contains at least one letter from $\overline{\Gamma}$, since $|w_4| > 2$.For w_2, w_3 and w_4 there exist splittings such that $w_2 = w_2' w_2''$, $w_3 = w_3' w_3''$ and $w_4 = w_4' w_4''$ with $w_2'' w_3' = \sharp\sharp$ and $w_3'' w_4' = \sharp\sharp$. Since both w_2 and w_4 are nonempty, we know that there exists an $i, 1 \leqslant i < |u|$ with $w_2' = \overline{a}_1 \sharp\sharp \cdots \overline{a}_i$ and $w_4'' = \overline{a}_{i+1} \sharp\sharp \cdots \overline{a}_{|u|} \sharp\sharp$. Now there are three sub-cases, depending on the length of w_1, w_2'', and i:
Let $k := |v| - |u| + i$. If $k > 0$ this will be used to calculate a split point for the compressed word which is to be substituted. In some cases we use a new nonterminal ξ_t, which will be added to Γ_2. Let the weight of ξ_t be $\psi(\xi_t) := \psi(\xi_{w_4 w_5}) - 1$.

(a) If $k \leqslant 0$ and $w_1 = \square$ add the rule

$$r_t := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(b) If $k \leqslant 0$ and $w_1 \neq \square$ add the following three rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{\sharp\sharp(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(c) If $k > 0$ add:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{|v|})\sharp\sharp w_5}w).$$

9. For $n = 5$, $v \neq \square$, $w_3 \notin \{\sharp, \sharp\sharp\}$ (w_3 cannot be empty), and $w_4 \notin \{\square, \sharp, \sharp\sharp\}$ we will have the biggest collection of sub-cases. In this case each of the words w_2,w_3, and w_4 contains at least one letter from $\overline{\Gamma}$.For w_2, w_3 and w_4 there exist splittings such that $w_2 = w_2'w_2''$, $w_3 = w_3'w_3''w_3'''$ and $w_4 = w_4'w_4''$ with $w_2''w_3' = \sharp\sharp$ and $w_3'''w_4' = \sharp\sharp$. Since all w_2, w_3 and w_4 are nonempty, we know that there exist $i, j, 1 \leqslant i < j < |u|$ with $w_2' = \overline{a}_1\sharp\sharp\cdots\overline{a}_i$, $w_3'' = \overline{a}_{i+1}\sharp\sharp\cdots\overline{a}_j$, and $w_4'' = \overline{a}_{j+1}\sharp\sharp\cdots\overline{a}_{|u|}\sharp\sharp$. Now there are 23 sub-cases(!), depending on the length of w_1, w_2'', w_3''', i, and j:
Let $k := |v| - |u| + i$. If $k > 0$ this will be used to calculate a split point for the compressed word which is to be substituted. Similarly, we will use $l := |v| - |u| + j$. Note that $l > 0$ implies $|v| \geqslant 2$. In some cases we use a new nonterminal ξ_t, which will be added to Γ_2. Let the weight of ξ_t be $\psi(\xi_t) := \psi(\xi_{w_4w_5}) - 1$.

(a) $k \leqslant 0$, $l \leqslant 0$, and $w_1 = \square$. We only use a single rule:

$$(\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{(\overline{b_1}\sharp\sharp)(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(b) $k \leqslant 0$, $l > 0$, $w_1 = \square$, and $w_3''' = \square$. We add three rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_l)\sharp}\xi_t w)$$
$$r_{t,3} := (\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_l)\sharp}\xi_t w, \xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_l)\sharp}\xi_{\sharp(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(c) $k \leqslant 0$, $l > 0$, $w_1 = \square$, and $w_3''' \neq \square$. Again, add three rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_t w)$$
$$r_{t,3} := (\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_t w, \xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_{(\sharp\sharp b_{l+2})\cdots(\sharp\sharp b_{|v|})\sharp\sharp w_5}w).$$

(d) $k \leqslant 0$, $l \leqslant 0$, and $w_1 \neq \square$. We will have three new rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{(\overline{b}_1\sharp\sharp)(b_2\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(e) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \square$, and $w_3''' = \square$. Add four rules:

$$r_{t,1} := (\xi_{w_1 w_2}\xi_{w_3}\xi_{w_4 w_5}w, \xi_{w_1 w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1 w_2}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{j-i})\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{j-i})\sharp}\xi_t w, \xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{j-i})\sharp}\xi_{\sharp(b_{j-i+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

Note that in this case it is easier to use $j - i$, because the first compression letter of the result contains no index letter from $\overline{\Gamma}$.

(f) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \sharp$, and $w_3''' = \square$. Add four rules:

$$r_{t,1} := (\xi_{w_1 w_2}\xi_{w_3}\xi_{w_4 w_5}w, \xi_{w_1 w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1 w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{\sharp(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1}\xi_{\sharp(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{\sharp(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)}\xi_{\sharp(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(g) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \sharp\sharp$, and $w_3''' = \square$. Add four rules:

$$r_{t,1} := (\xi_{w_1 w_2}\xi_{w_3}\xi_{w_4 w_5}w, \xi_{w_1 w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1 w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)}\xi_{\sharp(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(h) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \square$, and $w_3''' = \sharp$. Add four rules:

$$r_{t,1} := (\xi_{w_1 w_2}\xi_{w_3}\xi_{w_4 w_5}w, \xi_{w_1 w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1 w_2}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{j-i})\sharp\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{j-i})\sharp\sharp}\xi_t w, \xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{j-i})\sharp\sharp}\xi_{(b_{j-i+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

Again, it is easier to use $j - i$, because the first compression letter of the result contains no index letter from $\overline{\Gamma}$.

(i) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \sharp$, and $w_3''' = \sharp$. Add four rules:

$$r_{t,1} := (\xi_{w_1 w_2}\xi_{w_3}\xi_{w_4 w_5}w, \xi_{w_1 w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1 w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{\sharp(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1}\xi_{\sharp(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{\sharp(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_{(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(j) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \sharp\sharp$, and $w_3''' = \sharp$. Add four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_{(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(k) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \square$, and $w_3''' = \sharp\sharp$. Add four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1}\xi_{w_3}\xi_t w, \xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_t w, \xi_{w_1}\xi_{\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_{\sharp\sharp(b_{l+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

Here, we do not use $j - i$ although the first compression letter of the result contains no index letter from $\overline{\Gamma}$, because the middle compression letter of the result ends with a letter from Γ.

(l) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \sharp$, and $w_3''' = \sharp\sharp$. Add four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1}\xi_{(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_t w, \xi_{w_1\overline{b}_1}\xi_{(\sharp\sharp b_2)\cdots(\sharp\sharp b_{l+1})}\xi_{\sharp\sharp(b_{l+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(m) $k \leqslant 0$, $l > 0$, $w_1 \neq \square$, $w_2'' = \sharp\sharp$, and $w_3''' = \sharp\sharp$. Add four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_{l+1}\sharp)}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_{l+1}\sharp)}\xi_t w, \xi_{w_1\overline{b}_1\sharp}\xi_{(\sharp b_2\sharp)\cdots(\sharp b_{l+1}\sharp)}\xi_{\sharp(b_{l+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(n) $k > 0$, $l > 0$, $w_2'' = \square$, $w_3''' = \square$. Again, we use four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp}\xi_{\sharp(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(o) $k > 0$, $l > 0$, $w_2'' = \square$, $w_3''' = \sharp$. Add the following rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp\sharp}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp\sharp}\xi_{(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(p) $k > 0$, $l > 0$, $w_2'' = \Box$, $w_3''' = \sharp\sharp$. Add the following rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_k)\sharp\sharp}\xi_{b_{k+1}(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_{\sharp\sharp(b_{l+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(q) $k > 0$, $l > 0$, $w_2'' = \sharp$, $w_3''' = \Box$. Again, we use four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp}\xi_{\sharp(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(r) $k > 0$, $l > 0$, $w_2'' = \sharp$, $w_3''' = \sharp$. Again, we use four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp\sharp}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_l)\sharp\sharp}\xi_{(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(s) $k > 0$, $l > 0$, $w_2'' = \sharp$, $w_3''' = \sharp\sharp$. Again, we use four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})}\xi_{(\sharp\sharp b_{k+2})\cdots(\sharp\sharp b_{l+1})}\xi_{\sharp\sharp(b_{l+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(t) $k > 0$, $l > 0$, $w_2'' = \sharp\sharp$, $w_3''' = \Box$, $w_3 \notin \overline{\Gamma}$. Note that under these premises $l - k > 1$. Again, we use four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)}\xi_{\sharp(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(u) $k > 0$, $l > 0$, $w_2'' = \sharp\sharp$, $w_3''' = \Box$, $w_3 \in \overline{\Gamma}$. Note that in this case $k + 1 = l$. This case only uses three rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{\sharp(b_{k+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(v) $k > 0$, $l > 0$, $w_2'' = \sharp\sharp$, $w_3''' = \sharp$. Again, we use four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)\sharp}\xi_{(b_{l+1}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

(w) $k > 0$, $l > 0$, $w_2'' = \sharp\sharp$, $w_3''' = \sharp\sharp$. Again, we use four rules:

$$r_{t,1} := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4w_5}w, \xi_{w_1w_2}\xi_{w_3}\xi_t w)$$
$$r_{t,2} := (\xi_{w_1w_2}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w)$$
$$r_{t,3} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{w_3}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)\sharp b_{l+1}}\xi_t w)$$
$$r_{t,4} := (\xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)\sharp b_{l+1}}\xi_t w, \xi_{w_1\overline{b}_1(\sharp\sharp b_2)\cdots(\sharp\sharp b_{k+1})\sharp}\xi_{(\sharp b_{k+2}\sharp)\cdots(\sharp b_l\sharp)\sharp b_{l+1}}\xi_{\sharp\sharp(b_{l+2}\sharp\sharp)\cdots(b_{|v|}\sharp\sharp)w_5}w).$$

10. If $n > 5$ we only compress the information. In consequence, any reduction will take place by the rules of the cases for $n \leqslant 5$.
Consider all $t = (u, v, \xi_{w_1w_2}\xi_{w_3}\xi_{w_4}\cdots\xi_{w_{n-2}}\xi_{w_{n-1}w_n}w) \in T_2$ with $n \geqslant 6$. Then $\xi_{w_{n-3}w_{n-2}} \in \Gamma_2$. So, add the rule:

$$r_t := (\xi_{w_1w_2}\xi_{w_3}\xi_{w_4}\cdots\xi_{w_{n-2}}\xi_{w_{n-1}w_n}w, \xi_{w_1w_2}\xi_{w_3w_4\cdots w_{n-2}}\xi_{w_{n-1}w_n}w).$$

References

[1] M. Beaudry, M. Holzer, G. Niemann, and F. Otto. McNaughton languages. Technical Report Mathematische Schriften Kassel, No. 26/2000, Universität Kassel, November 2000. Full version of [2].

[2] M. Beaudry, M. Holzer, G. Niemann, and F. Otto. McNaughton languages. In: R. Freund (Ed.), Theorietag 2000 mit Workshop New Computing Paradigms: Molecular Computing and Quantum Computing. Technische Universität Wien, 2000, pp. 159–165. ISBN 3-85028-325-9. Extended abstract of [1].

[3] R.V. Book, Confluent and other types of Thue systems, JACM 29 (1982) 171–182.

[4] R.V. Book, F. Otto, String-Rewriting Systems, Springer-Verlag, New York, 1993.

[5] G. Buntrock, Wachsende kontext-sensitive Sprachen, Habilitationsschrift, Fakultät fr Mathematik und Informatik, Universität Würzburg, Juli 1996.

[6] G. Buntrock, F. Otto, Growing context-sensitive languages and Church–Rosser languages, Information and Computation 141 (1998) 1–36.

[7] N. Chomsky, On certain formal properties of grammars, Information and Control 2 (2) (1959) 137–167.

[8] E. Dahlhaus, M.K. Warmuth, Membership for growing context-sensitive grammars is polynomial, Journal of Computer and System Sciences 33 (1986) 456–472.

[9] M.A. Harrison, Introduction to Formal Language Theory, Addison-Wesley, Reading, MA, 1978.

[10] M. Jantzen, Confluent String Rewriting, Springer-Verlag, Berlin, 1988.

[11] D. Kapur, M. Krishnamoorthy, R. McNaughton, P. Narendran, An $O(|T|^3)$ algorithm for testing the Church–Rosser property of Thue systems, Theoretical Computer Science 35 (1985) 109–114.

[12] R. McNaughton, An insertion into the Chomsky hierarchy?, in: J. Karhumäki, H.A. Maurer, G. Paŭn, G. Rozenberg (Eds.), Jewels are Forever, Contributions on Theoretical Computer Science in Honour of Arto Salomaa, Springer-Verlag, Berlin, 1999, pp. 204–212.

[13] R. McNaughton, P. Narendran, F. Otto, Church–Rosser Thue systems and formal languages, Jounral of Association Computing Machinery 35 (1988) 324–344.

[14] G. Niemann, F. Otto, The Church–Rosser languages are the deterministic variants of the growing context-sensitive languages, in: M. Nivat (Ed.), Proc. of FoSSaCS 1998 (Foundations of Software Science and Computation Structures), Lecture Notes on Computer Science, vol. 1378, Springer-Verlag, Berlin, 1998, pp. 243–257.
[15] G. Niemann and J.R. Woinowski, The growing context-sensitive languages are the acyclic context-sensitive languages. In: W. Kuich (Ed.), Preproc. of DLT 2001 (Developments in Language Theory). Institut für Algebra und Mathematik, TU Wien, 2001.
[16] J.R. Woinowski, A normal form for Church–Rosser language systems. Technical Report TI-7/00, TU-Darmstadt, June 2000.
[17] J.R. Woinowski, A normal form for Church–Rosser language systems, in: Aart Middeldorp (Ed.), Proc. of RTA 2001 (Rewriting Techniques and Applications), Lecture Notes on Computer Science, vol. 2051, Springer-Verlag, Berlin, 2001, pp. 322–327.