On String Rewriting Systems
— DRAFT —

Nicolai Kraus and Christian Sattler

January 13, 2012

Abstract

String rewriting systems, also known as semi-Thue systems, consist of a set of rules $l \to r$, specifying valid replacements of substrings of strings over a given alphabet. In the case of one single rule, it is an open problem whether there is a system that is neither terminating nor looping. Another open question is the decidability of termination. Difficulties arise especially for non-confluent systems.

In this article, we develop strategies for approaching non-confluent string rewriting in general, not necessarily requiring only one rule. For a given system, we define an equivalent system with confluence-like properties, enabling us to develop some normal form theorems.

Subsequently, we apply our methods to present a partial solution to the first of the mentioned problems, going one step further than Geser's previous result by only requiring that there is a unique overlap in one order, but not making any restrictions for the other order. However, one case, where the right side is mainly periodic with a very special period, remains open. We further discuss implications for the latter question.

We believe that our approach can be used to attack other questions in the subject of string rewriting.

1 Introduction and Related Work

General stuff: finite is enough; 01 is enough; number or ratio of 0,1 occurrences is difficult to use.

Approaches (for the single rule case) so far: Geser, Kurth, Zantema, Kobayashi et al, McNaughton, ...

Also mention *Open. Closed. Open.* and the RTA information. Links: Decidability of Termination, Termination and Looping problem.

Cite the following: [1], [2], [3], [4], [5], [6], [7], to be extended.

2 Preliminaries

In our notations, we follow [2] and others. As usual, an *alphabet* Σ is a set (we do not make restrictions on the size) the elements of which we call *letters*. We also have Σ^*, the set of *words* or *strings* over Σ. In this paper, we prefer the latter term. For strings s, t we write st for the obvious composition and $|s|$ resp. $|t|$ for the length. A string s is called a *prefix* of t if there exists some u with

$t = su$, or, analogously, a *suffix* if $t = us$. We want to explicitly exclude the possibility that u is the empty word ϵ. Furthermore, s is a *factor* ot t if there are nonempty words u, v with the property $t = usv$. Note that s can be a prefix, suffix and factor at the same time. We define the set of *overlaps* of a string u with a string v by

$$OVL(u, v) := \{w \in \Sigma^+ | u = u'w, v = wv', \textit{where } u', v' \in \Sigma^*, u'v' \neq \epsilon\}.$$

A subset of $\Sigma^* \times \Sigma^*$ is (together with Σ) called a *string rewriting system*, a single element a *string rewriting rule*. Such a system R induces a relation $\to_R \subseteq \Sigma^* \times \Sigma^*$, where $u \to_R v$ if and only if there is a rule $(l, r) \in R$ and $p, q \in \Sigma^*$ satisfying $u = plq$ and $v = prq$.

The System (Σ, R) *terminates* if there is no infinite sequence $s_1 \to_R s_2 \to_R \ldots$. Further, it is said to *loop* if there are strings $s, u, v \in \Sigma^*$ satisfying $s \to_R^+ usv$. We denote the set of all possible sequences by Seq^R. The set of all sequences, together with precise information which rule has been used at which position for every step, is called Seq_0^R. There is an obvious surjection $Seq_0^R \twoheadrightarrow Seq^R$. Further, we can view Seq^R and Seq_0^R as categories, where objects are strings over Σ and a morphism $w \xrightarrow{f} w'$ is a finite sequence starting with w and ending in w', which, in the case of Seq_0^R, carries information on the steps and applied rules. The mentioned surjection becomes a surjective functor.

A sequence has the *rightmost rewriting property* if, at each step, the rightmost substring (the substring defined by positions $i < j$ with the least possible i) that matches a left side of a rule is rewritten.

For sets, we always use capital identifiers, while words are denoted using lowercase. For a word w, the notation w^* and w^+ are used in the obvious sense, namely $\{w\}^*$ and $\{w\}^+$. Note that, consequently, they are sets, not single words; on the other hand, for a natural number k, w^k is a single word, namely the word $ww \ldots w$.

As a letter is just an element of an alphabet, we speak of *symbols* whenever me mean a very concrete letter at a concrete position in a string.

As usual, we use $\pi_1 : A \times B \times \ldots \to A$, π_2, π_3 and so on for the first, second and third projection.

Assume A is a set and $\perp$ is a distinguished character TODO, not contained in A. By $A_\perp$, we simply denote the set $A \cup \{\perp\}$. Further, if B is also a set, given a function $f : A \to B$, we write $\overset{\perp}{f}: A_\perp \to B_\_g$ for the function that is defined by simple extending the domain of f by setting $\overset{\perp}{f}(\perp) = \perp$.

We use the function $map : (A \to B) \to A^* \to B^*$ in the obvious meaning, $map\ f(a_1a_2 \ldots a_k) = f(a_1)f(a_2) \ldots f(a_k)$. In the same way, we define the function $maps : (A \to B^*) \to A^* \to B^*$, composing strings instead of single letters on the right hand side of the definition.

For the next sections, we assume that Σ is an alphabet and R a string rewriting system over Σ. We only consider finite set of rules, so we set $R = \{(l_i, r_i) | 1 \leq i \leq n\}$ for some $n > 0$.

3 The Oracle Translation

In this section, we describe a transformation, yielding a new rewriting system over another alphabet which is (in a very strong sense) equivalent to R. A similar approach was done in [6]. However, their idea crucially depends on the confluence of the original one-rule rewriting system, while our algorithm transforms any string rewriting system into one without any non-complete overlapping of left sides. We want to admit, though, that our approach is straightforward and we would not be surprised if it had already been used before, but we are not aware of anyone who has done it.

We first give the definition and try to provide an intuitive explanation afterwards.

Definition 1 (cool sounding name translation, e.g. Oracle Translation, or just canonical translation / annotation)**.** *Given a finite alphabet Σ and a finite number of rules $R = \{(l_1, r_1), \dots, (l_n, r_n)\}$, we define a new (finite) alphabet $\Sigma^{\neptune}$ by*

$$\Sigma^{\neptune} := (\Sigma \times \{s, m, e\} \times \{1, 2, \dots, n\})_\perp$$

together with a new finite set of rules

$$R^{\neptune} := \{(l, r) \in \Sigma^{\neptune} \times \Sigma^{\neptune}$$

such that there is a $k \in \{1, 2, \dots, n\}$ where:

- *map* $\overset{\perp}{\pi_1}\,(l) = l_k$
- *map* $\overset{\perp}{\pi_2}\,(l) = s\underbrace{mm\dots m}_{|l_k|-2}e$
- *map* $\overset{\perp}{\pi_3}\,(l) = \underbrace{kk\dots k}_{|l_k|}$
- *regarding map* $\overset{\perp}{\pi_1}$ *r, if we replace the first occurrence of $\perp$ by z_1, the second by z_2 and so on, all $z_i \notin \Sigma$, we get a word $w \in (\Sigma \cup \{z_1, \dots, z_m\})^*$ which satisfies $\exists z_1, \dots, z_m \in \Sigma . w = r_k\}$*

While this definition might be irritating at first sight, all we do is adding some additional information to the symbols, restricting the way in which they can be used later. s, m, e stand for *start*, *mid*, *end*. We think of an element of $\Sigma \times \{s, m, e\} \times \{1, 2, \dots, n\}$ as a letter from the original alphabet, annotated with k if this symbol will be used using the k^{th} rewriting rule, and, moreover, s, e or m, meaning that the symbol will be the first resp. the last resp. one of the symbols in the middle of l_k. Finally, $\perp$ is *garbage* in the sense that this symbol can never be used again. Note that the definition makes it impossible for $\perp$ do occur on the left side of a rule.

We define a sequence over $(\Sigma^{\neptune}, R^{\neptune}$ to be *valid* if everything that is not used is declared as garbage:

Definition 2 (Validity of sequences)**.** *A sequence $s_0 \to_{R^{\neptune}} s_1 \to_{R^{\neptune}} \dots \in Seq_0^{R^{\neptune}}$, finite or not, is called* valid *if, for every p and every symbol α of s_p which is not equal to $\perp$, there is a step p' such that α does not appear in the string $s_{p'}$ (note that another symbol, equal to α, may very well appear). More precisely, only symbols that equal $\perp$ do exist infinitely long, and, if the sequence is finite, it ends with $\perp\perp\dots\perp$. We use the notation $ValSeq_0^R$ for the set of all valid sequences.*

Remark 3. *While this definition makes sense insofar as that it provides canonical forms for sequences. However, at the same time, it makes the categorical view difficult as composition of sequences will not be possible in the way it should... the author is therefore not sure about these issues.*

There is a simple, however crucial, fact to mention about this:

Theorem 4. *There is a bijection $\phi : Seq_0^R \to ValSeq_0^R$ which is natural in the sense that, if $s = s_0 \to s_1 \to \dots$ and $\phi(s) = s'_0 \to s'_1 \to \dots$, then map $\pi_1(s'_i)$ has, on every position, either $\perp$ or the same letter as s_i.*

Proof. This is obvious from the definition. □

Remark 5. *Our translation $\_^{\text{⯉}}$ can be seen as the* cool sounding name of level 1*, as we include, for every symbol, information on how it will be used in its next step. However, we could also include information on what happens to the word it is rewritten to in that step. As this word consists of (usually) multiple symbols, we would have to state the future for each of these symbols. In general, in the translation of level k, each symbol would be annotated with a tree of height k, where the root (which, in our understanding, has height 1) determines the way that the symbol is rewritten, the children of the root determine the way that the children of the symbol are rewritten, and so on. While this construction has lots of nice and interesting properties, we do not see how it could be used, so we restrict ourselves to level 1. It is worth mentioning that, at level ∞, everything is predetermined and all questions regarding looping, termination and so on have equivalent formulations talking about trees.*

Our freshly constructed rewriting system has the following property:

Proposition 6. *If (l_1, r_1) and (l_2, r_2) are two (not necessarily different) rules in $R^{\text{⯉}}$, l_1 and l_2 are completely overlapping free, i.e. $OVL(l_1, l_2) = OVL(l_2, l_1) = \emptyset$. Moreover, none is a factor of the other.*

Proof. This is an immediate consequence of the fact that the second projection of every left side is $smm \dots me$. □

TODO define ancestor, child...

Unfortunately, the above property does not imply confluence. This would only be the case if not two left sides were equal, but in all interesting cases, they are not. However, a weak from of confluence or reordering [TODO figure out how this is called!] holds:

Proposition 7. *Given two sequences $s_0 \to s_1 \to s_2 \to \dots$ and $s_0 \to s'_1 \to s'_2 \to \dots$, both in $ValSeq_0^R$, finite or not, with the same starting string. Let k, k' be two natural numbers, not larger then the length of the first resp. the second sequence. For every symbol x of s_0, let F_x resp. F'_x be the future of x until step k resp. k'. If, for all symbols x, it is the case that F_x is a of F'_x or F'_x is a prefix of F_x or they are equal, then it is possible to find a t such that $s_k \to^* t$ and $s'_k \to^* t$.*

Proof. Clear. □

4 Normal Forms

At this point, we want to push the level of abstraction one level higher. While ValSeq_0^R has very nice properties, our translation can still be improved. As a motivation for our next step, assume t is a string occurring in a valid sequence. Then, *map* $\overset{\perp}{\pi_2}$ (s_i) is any string over $\{s, m, e, \perp\}^m$, for example, $\perp\perp\perp smmmsse\perp\perp smmese\perp\perp\perp$. We know that the sequence is valid, so every symbol except $\perp$ will, at some point, be rewritten. At the same time, we know that only $sm \dots me$ can be rewritten. This has some immediate consequences, for example, there can never be an e after a $\perp$ as there will be no way to insert the necessary s between them. But, more important at the moment, we note that some symbols will, not matter what we do, be rewritten together. If we have two symbols mm, we know that those will be rewritten together, as there is no rule allowing us to insert an s in between. This observation allows us to group symbols and build blocks.

Definition 8. *Let Σ, R be a string rewriting system with n rules as above. We define a new alphabet $\Sigma^{\mars}$ by*

$$\Sigma^{\mars} := \{(w, f, k) \text{ where } f \in \{S, M, E\}, 1 \leq k \leq n, w \in \Sigma^* \text{ such that,}$$

if $f = S$, then w is a prefix, if $f = E$, a suffix, and if $f = M$, a factor of $l_k\}_\perp$ with corresponding rules

$$R^{\mars} := \{(l, r)\} \subseteq \Sigma^{\mars *} \times \Sigma^{\mars *}$$

such that there is a $k \in \{1, 2, \dots, n\}$ where:

- *maps* $\overset{\perp}{\pi_1}$ $(l) = l_k$
- *maps* $\overset{\perp}{\pi_2}$ $(l) = SMM \dots ME$
- *maps* $\overset{\perp}{\pi_3}$ $(l) = kk \dots k$
- *regarding maps* $\overset{\perp}{\pi_1}$ r, *if we replace the first occurrence of $\perp$ by z_1, the second by z_2 and so on, all $z_i \notin \Sigma$, we get a word $w \in (\Sigma \cup \{z_1, \dots, z_m\})^*$ which satisfies $\exists z_1, \dots, z_m \in \Sigma^*. w = r_k\}$*

Remark 9. *Of course, there is an obvious inclusion $R^{\mercury} \hookrightarrow R^{\mars}$ and a surjection $R^{\mars} \twoheadrightarrow R^{\mercury}$, the first being a section of the latter. Note that there is also another canonical map $\phi : R^{\mercury} \to R^{\mars}$, the "greedy" one, which always tries to build blocks as large as possible. This is the function we are actually interested in:*

Definition 10 ("Greedy block building function", find better name)**.** *$\phi : \Sigma^{\mars *} \to \Sigma^{\mars *}$ is the function defined by:*

$$\begin{aligned}
\phi(w, f, k) &= (w, f, k) \\
\phi((w, s, k_1)(v, m, k_1)a_3 \dots a_x) &= \phi((wv, s, k_1)a_3 \dots a_x) \\
\phi((w, m, k_1)(v, m, k_1)a_3 \dots a_x) &= \phi((wv, m, k_1)a_3 \dots a_x) \\
\phi((w, m, k_1)(v, e, k_1)a_3 \dots a_x) &= \phi((wv, e, k_1)a_3 \dots a_x) \\
\phi(a_0 a_1 \dots a_x) &= a_0 \phi(a_1 \dots a_x)
\end{aligned}$$

Further, define a minimal set of rules (which make still every sequence possible that is possible with $R^{\overset{\uparrow}{\odot}}$) by $R^{\overset{\uparrow}{\odot}}{}_{min}$. We further require that ϕ is, on all right sides of $R^{\overset{\uparrow}{\odot}}{}_{min}$, the identity. Note that this is possible. Also note that such only the rules that "make sense" in the sense of "producing valid sequences". TODO

A key key property is the following:

Theorem 11. *If r is a right side of a rule in $R^{\overset{\uparrow}{\odot}}{}_{min}$, then we have*

$$map\ \overset{\perp}{\pi_2}\ (r) \in e^*m?s^*.$$

Proof. Straightforward. □

We define the set $\mathfrak{S}(R)$ to be the set of valid infinite rightmost sequences were every (produced) non-$\perp$ symbol has an infinite trace. It is easy to show that there is only one "block" of non-$\perp$-symbols.

Lemma 12. *If R allows the existence of infinite sequences, then $\mathfrak{S}(R)$ is nonempty.*

Proof. Needs lemma: finite and infinte trees do not interact. Clear. □

Remark 13. *Our original proofs included switching between left- and rightmost rewriting regularly, forcing symbols to become $\perp$ at every switch if they would be delayed infinitely long. This approach should be examined further.*

Lemma 14. *For every string occurring in a sequence in $\mathfrak{S}(R)$, there is only one "connected" block of non-$\perp$ symbols.*

Proof. Guaranteed by the rightmost rewriting property. □

Definition 15. *Take $\mathfrak{S}(R)$, apply ϕ, for every sequence, on every word. The result is the set $\mathfrak{I}(R)$ of "everywhere maximal blocked, rightmost, valid, infinite" sequences. These are not "real" sequences though. TODO!*

Corollary 16. *If R does not terminate, this set is nonempty.*

Theorem 17. *In every sequence of this set, there are , for some k, infinitely many occurrences which are also in $\perp?l_ke^*\perp?$.*

This is our left normal form. The right normal form is the corresponding "dual".

Proof. For every s_i, there is a rightmost symbol that is not $\perp$. Because of our definitions, this symbol will be rewritten at some point. Precisely before this point, we obviously have the required form □

Remark 18. *Because of compactness, we may assume that a subsequence of these normal forms converges (with respect to the canonical metric), the limit being something in $\perp?l_ke^\infty$. We are not sure whether this is of any use.*

Remark 19. *Applying the argument that we can, after reaching a normal form, change from rightmost to leftmost rewriting, it is easy to see that we can get a subsequent where every odd entry is a left normal form, every even is a right one.*

5 One-Rule System with only one overlap

In this section, we want to have a closer look at string rewriting systems with only one rule. This case is particularly interesting as it is a long standing open problem whether, given a single rule, uniform termination is decidable. ET CETERA. For more background, see our references. ET CETERA.

Question 20 (Decidability of termination)**.** *Given a single rule* (l, r)*, is it decidable whether there is an infinite sequence* $s_0 \to s_1 \to \ldots$*?*

Let us have a look on what can happen. For example, given the rule (ab, bbb), we have the sequence $abbaba \to bbbbaba \to bbbbbbbba$, which obviously cannot be continued. It is easy to see that an infinite sequence cannot exist, as in every step, we reduce the number of a's by one. For a different example, look at the rule $(aab, bbaaaa)$, which enables us to build the sequence $\overline{aab}b \to bbaa\overline{aab} \to bb\underline{aabb}aaaa$. Noting that the last word contains the starting string of the sequence, it is clear that we can use this loop to construct an infinite sequence. It is, however, not clear whether there is a third possibility.

Question 21. *Is every single rule string rewriting system either terminating or looping?*

Both of these questions are open and have attracted quite a bit of interest. The first was originally formulated in 1991? by Max Dauchet?, and, 2? years later, added to the open problems list of *Rewriting Techniques and Applications (RTA)*. The second, posed by Hans Zantema? in 1999?, is also contained in the RTA list.

Many attempts have been made and a lot of partial solutions have been published. Notable is, for example, the result that both questions can be answered positively if the left side of the rule is of the form $l = a^p b^q$, though this case has the advantage of inducing a confluent system as $OVL(l, l) = \emptyset$. Other people (TODO add names and precise results) have looked at the sets $OVL(l, r)$ and $OVL(r, l)$ instead. It is quite easy to see that the system will always terminate, provided that one of those is empty. Alfons Geser has solved the case that both sets have exactly one inhabitant completely, also answering both questions positively. In this section, we want to go one step further, only assuming that one of the two sets is a singleton, but not making any restrictions for the other.

Theorem 22. *Given a rule* (l, r)*, suppose we have* $OVL(r, l) = \{u\}$*. By* v*, denote the word that satisfies* $r = uv$*. Then, if* r *is not of the form* $xv^k y$*, where* k *is any natural number (possibly* 0*),* x *is a suffix of* r*, and* y *is a prefix of* v*, the system is either termination or looping.*

Remark 23. *The conditions seem to be quite weird, but we basically just assume that the right hand side is not mainly periodic with the period being the complement of the single overlapping. We hope that we can weaken these assumption.*

Proof. By $\psi : \Sigma^{\overset{\uparrow}{\odot}\,*} \to \mathbb{N}$, denote the number of end-blocks that are not equal to v (not counting those occurring in redexes). Whenever we have a left normal form, we apply leftmost rewriting until we reach a right normal form, where we switch back to rightmost rewriting again. Obviously, ψ is 0 for right normal forms. Note that, provided that r is not of the special form stated in the theorem, there is no step that reduces ψ, thereby forcing ψ to be zero for every left normal form, which leads to an obvious loop (or termination, if only finitely many times a normal form is reached). □

Remark 24. *Of course, we should also discuss decidability of termination (we can get a result that is even weaker than the above easily). It should be possible to get Geser's [2] result using this strategy (but the author is not sure).*

6 Conclusions and Future Work

More ideas:

christian's strategy: use new redices and build a system without special pairs. higher order redices.

there is also the special-pair-criterion. no special pair: trivial, ie BmB never has an A: trivial.

higher order redex: AmAmAmAmA

only finitely many of these redexes: solvable! (I think)

etc.

References

[1] Nachum Dershowitz. "Open. Closed. Open". In: *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings.* Ed. by Jrgen Giesl. Vol. 3467. Lecture Notes in Computer Science. Springer, 2005, pp. 376–393. ISBN: 3-540-25596-6. DOI: `http://springerlink.metapress.com/openurl.asp?genre=article&amp;issn=0302-9743&amp;volume=3467&amp;spage=376`.

[2] Alfons Geser. "Termination of string rewriting rules that have one pair of overlaps." English. In: Berlin: Springer, 2003.

[3] Winfried Kurth. *Termination und Konfluenz von Semi-Thue-Systemen mit nur einer Regel. (Termination and confluence of semi-Thue-systems with a single rule).* German. Clausthal: Techn. Univ. Clausthal, Math.-Naturwiss. Fak., 1990.

[4] Hans Zantema and Alfons Geser. *Non-Looping Rewriting.* 1996.

[5] H. Zantema and A. Geser. "A Complete Characterization of Termination of $0^p1^q \to 1^r0^s$". In: *Rewriting Techniques and Applications.* Ed. by J. Hsiang. Berlin, Heidelberg: Springer, 1995, pp. 41–55.

[6] Yuji Kobayashi, Masashi katsura, and Kayoko Shikishima-Tsuji. "Termination and derivational complexity of confluent one-rule string-rewriting systems". In: *Theor. Comput. Sci.* 262 (1-2 2001), pp. 583–632. ISSN: 0304-3975. DOI: `http://dx.doi.org/10.1016/S0304-3975(00)00367-4`. URL: `http://dx.doi.org/10.1016/S0304-3975(00)00367-4`.

[7] Robert Mcnaughton. *Well behaved derivations in one-rule semi-Thue systems.* 1995.