Effective Lossy Queue Languages

Parosh Aziz Abdulla[1], Luc Boasson[2], and Ahmed Bouajjani[2] *

[1] DoCS, Uppsala University, P.O. Box 325, S-75105 Uppsala, Sweden.
parosh@docs.uu.se
Phone: (+46) 18 471 31 63. Fax: (+46) 18 55 02 25.

[2] Liafa, Univ. of Paris 7, Case 7014, 2 place Jussieu, F-75251 Paris Cedex 05, France.
{lub,abou}@liafa.jussieu.fr
Phone: (+33) 1 4427 2841/7819. Fax: (+33) 1 4427 6849.

ICALP'01 submission, TRACK B

Abstract. Although the set of reachable states of a lossy channel system (LCS) is regular, it is well-known that this set cannot be constructed effectively. In this paper, we characterize significant classes of LCS for which the set of reachable states can be computed. Furthermore, we show that, for slight generatlizations of these classes, computability can no longer be achieved.
To carry out our study, we define *rewriting systems* which capture the behaviour of LCS, in the sense that (i) they have a FIFO-like semantics and (ii) their languages are downward closed with respect to the substring relation. The main result of the paper shows that, for context-free rewriting systems, the corresponding language can be computed. An interesting consequence of our results is that we get a characterization of classes of meta-transitions whose post-images can be effectively constructed. These meta-transitions consist of sets of nested loops in the control graph of the system, in contrast to previous works on meta-transitions in which only single loops are considered.
Essentially the same proof technique we use to show the result mentioned above allows also to establish a result in the theory of $0L$-systems, i.e., context-free parallel rewriting systems. We prove that the downward closure of the language generated by any $0L$-system is effectively regular.

Keywords: Verification, Infinite-state systems, Rewriting systems, Language theory.

* Corresponding author.

1 Introduction

We consider the problem of model checking of *lossy channel systems (LCS)* which consist of finite-state processes communicating over FIFO-buffers. The buffers are unbounded and *lossy* in the sense that they can lose messages. Such systems can be used to model the behaviour of communication protocols such as link protocols [AJ96] and bounded retransmission protocols [GvdP93,AAB99] which are designed to operate correctly even in the case where the underlying communication medium cannot provide reliable communication. In [AJ96], an algorithm for checking safety properties for LCS is described. The algorithm performs a fixed point iteration where each iteration computes the pre-image of a set of configurations with respect to a single transition of the system. This corresponds to a backward reachability analysis algorithm which constructs a characterization (as a regular set) of the set of configurations from which a set of *final configurations* is reachable.

Often, it is also important to be able to perform a forward reachability analysis, and characterize the set of configurations which are reachable from the initial configuration of the system. For instance, several efficient verification methods, such as on-the-fly algorithms [Hol91,CVWY90], are based are on forward search of the state space. Also, using forward analysis, we can often automatically generate finite-state abstractions of the system. The abstract model may then be used to check several classes of properties which cannot be analyzed directly in the original model. It is well known that the set of reachable configurations in an LCS can be characterized as a regular set [CFI96,ABJ98]. On the other hand, using undecidability results reported in [May00], it can be shown that this characterization cannot be effectively constructed. Therefore, it is interesting to characterize the class of LCS for which the characterization is effective.

To achieve that, we model the behaviour of an LCS as a rewriting system. To reflect the behaviour of the LCS, we provide a FIFO-like semantics for the rewriting system, i.e., when a rule is applied to a string, the left hand side is removed from the head of the string, while the right hand side is appended to the end of the string. Furthermore, corresponding to lossiness, we require that the language generated is downward closed with respect to the substring relation. Intuitively, the symbols in the left hand side of a rule correspond to the messages received from the channels, while the symbols in the right hand side correspond to messages sent to the channels. This implies that characterizing the class of rewriting systems with effectively computable languages gives also a characterization of the class of LCS with computable reachability sets.

In this paper, we study the limits of the computability of the reachability set, and characterize significant classes of rewriting systems which have effectively computable languages. The main result of the paper is showing that all context-free rewriting systems have effectively computable languages. Furthermore, we show that slight generalizations of the context-free class lead to uncomputability of the language. For instance, it is sufficient to allow two symbols in the left hand sides of the rewriting rules in order to lose computability. Moreover, if we consider rewriting systems on vectors of words (representing FIFO-channel systems with several queues), then computability is lost starting from dimension 2. Nevertheless, we show that for particular n-dim context-free rewriting systems called *rings*, the generated language is effectively computable. The class of rings corresponds to n-dim systems where each rewriting rule consists in receiving a message from some channel of index say i, and sending a string (sequence of messages) to the channel of index $(i + 1)$ mod n. These rules correspond to actions of communicating processes connected according to a ring topology, where each component has an input and an output buffer.

A significant application of our results is to characterize classes of *meta-transitions* which can be used in forward reachability analysis of LCS. A *meta-transition* is an *arbitrarily long sequence* of transitions, which often corresponds to repeated executions of loops in the syntactic code of the program. The idea is to speed up the iterations corresponding to forward reachability analysis by computing the post-image of a *meta-transition* rather than a single transition. In almost all practical cases, applying meta-transitions is necessary to achieve termination of the iteration procedure. Obviously, the key issue is to decide which meta-transitions to apply during each iteration step. A necessary criterion is that a meta-transition should allow computability of post-images, i.e., the post-image of each constraint with respect to the meta-transition should be computable. Since the set of transitions in a meta-transition can themselves be viewed as an LCS, our result also gives a characterization of meta-transitions which allow computing of post-images. Observe that the language generated describes the application of an arbitrary sequence of rules, which means that we are able to describe the effect of arbitrary sequences of control loops. In this way we extend significantly the class of LCS on which accelerations can be applied, since earlier works [BG96,BGWW97,BH97,ABJ98] only consider single loops.

It turns out that our proof can also be adapted to establish a result in the language theoretical framework of $0L$-systems [RS80]. These systems consist of context-free sets of rules with a parallel rewriting relation. In fact, there is a tight relation between FIFO rewriting and parallel rewriting (which is actually used in our proof). We show that the set of subwords of the language generated by any $0L$-system is an effective regular set.

Related Work: Several works have studied computing post-images of meta-transitions in the context of communicating finite-state processes [BG96,BGWW97,BH97,ABJ98]. In contrast to this paper, all these works consider meta-transitions which correspond to *single loops*. This excludes several common structures which occur in protocols such as *nested loops*. An example of a nested loop is two transitions sending two different messages to the channels. For instance, in [ABJ98], we show how to compute post-images of single loops for lossy channel systems. In [BGWW97], a complete characterization is given of the class of simple loops which preserve regularity in the context of systems with *perfect channels*. In [BH97] a constraint system is presented and shown to be closed under the execution of any single loop in a system with perfect channels.

Context-free rewriting systems have been used for description and analysis for several classes of infinite-state systems such as BPA and pushdown processes (e.g., [BBK93,CHS92,GH94,Sti96]). All these works deal with the classical (stack-like) semantics, and hence cannot be applied to analyze systems with FIFO buffers. [Bur98] considers rewriting systems with *perfect* FIFO behaviour. It shows that all interesting verification problems (including reachability) are undecidable for these systems.

$0L$-systems have been studied as an alternative of the classical sequential rewriting systems, following the main lines suggested by the theory of context-free grammars. It has been shown that the properties of the two kinds of formalisms are quite different. In particular, the expressive power of the $0L$-systems is not comparable with the usual Chomsky hierarchy. Our result concerning $0L$-system corresponds to the result established in [Cou91] for the context-free languages.

Outline: The remainder of the paper is organized as follows. In Section 2 we define some basic notions and introduce FIFO rewriting systems (which we call cyclic rewriting systems) and $0L$-systems. In Section 3 we prove our main result concerning the effective regularity of the language of downward closed context-free systems. We give also in this section our result concerning the effectiveness of the downward closure of $0L$ languages. In Section 4 we establish unconstructibility results for downward closed rewriting systems with unrestricted (non context-free) rules, as well as for 2-dim context-free systems. Finally, in Section 5, we prove the effective regularity of the n-dim context-free rings languages.

2 Preliminaries

In this section, we give some basic definitions and introduce the FIFO rewriting systems which we call *cyclic* rewriting systems.

2.1 Words and Languages

Let Σ be a finite alphabet. We denote by Σ^* the set of *words* over Σ, i.e., sequences (or strings) of symbols in Σ. We denote by ϵ the empty word.

We denote by $\preceq \subseteq \Sigma^* \times \Sigma^*$ the subword relation, i.e., $a_1 \ldots a_n \preceq b_1 \ldots b_m$ if there exists $i_1, \ldots, i_n \in \{1, \ldots, m\}$ such that $i_1 < \ldots < i_n$ and $\forall j \in \{1, \ldots, n\}.\ a_j = b_{i_j}$. We consider the product generalization of this relation to vectors of words.

An n-dim language over Σ, for $n \geq 1$, is any subset of $(\Sigma^*)^n$. 1-dim languages are called languages as usual. An n-dim language L is *downward closed* if $\forall \boldsymbol{u}, \boldsymbol{v} \in (\Sigma^*)^n$, if $\boldsymbol{v} \in L$ and $\boldsymbol{u} \preceq \boldsymbol{v}$, then $\boldsymbol{u} \in L$. Let $L\!\downarrow$ denote the *downward closure* of L, i.e., the smallest downward closed set which includes L. Notice that a set is downward closed if and only if $L = L\!\downarrow$.

2.2 Simple Regular Expressions

Let us call *atomic expression* any expression e of the form $(a + \epsilon)$ where $a \in \Sigma$, or of the form A^* where $A \subseteq \Sigma$. A *product* p is either the empty word ϵ, or a finite sequence of atomic expressions of the form $e_1 \cdots e_m$. A *simple regular expression* (SRE) is either $\emptyset$, or a finite union $p_1 + \cdots + p_n$ where each p_i is a product.

Theorem 1 ([ABJ98]). *SRE definable sets are precisely downward closed sets. Moreover, for every effective regular language L, $L\!\downarrow$ is an effective SRE language.*

Proposition 1 ([Cou91]). *For every context-free language L (effectively described by a context-free grammar), the set $L\!\downarrow$ is an effective regular set.*

2.3 Cyclic Rewriting Systems

For $n \geq 1$, an n-dim *rewriting rule* r over the alphabet Σ is a pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ where $\boldsymbol{x}, \boldsymbol{y} \in (\Sigma^*)^n$. We denote such a rule by $r : \boldsymbol{x} \mapsto \boldsymbol{y}$. The *left hand side* (resp. *right hand side*) of r, denoted by $lhs(r)$ (resp. $rhs(r)$), is the vector $\boldsymbol{x}$ (resp. $\boldsymbol{y}$).

An n-dim rewriting rule r is *context-free* if $lhs(r) \in (\Sigma \cup \{\epsilon\})^n$. A set rules R is context-free if all its rules are context-free.

An n-dim *cyclic rewriting system* is a pair (R, R_c) where R is a finite set of n-dim rewriting rules, and $R_c : (\Sigma^*)^n \to 2^{(\Sigma^*)^n}$ is the *cyclic rewriting mapping* over R defined as follows: for every $\boldsymbol{u} = (u_1, \ldots, u_n), \boldsymbol{v} = (v_1, \ldots, v_n) \in (\Sigma^*)^n$, we have $\boldsymbol{v} \in R_c(\boldsymbol{u})$ if and only if there exists a rule $r : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n) \in R$ such that $\forall i \in \{1, \ldots, n\}.\ \exists w_i \in \Sigma^*.\ u_i = x_i w_i$ and $v_i = w_i y_i$.

A *weak cyclic rewriting system* is a pair (R, R_{wc}) where R is a finite set of n-dim rules and R_{wc} is the *weak cyclic rewriting mapping* over R defined as follows: for every $\boldsymbol{u}, \boldsymbol{v} \in (\Sigma^*)^n$, we have $\boldsymbol{v} \in R_{wc}(\boldsymbol{u})$ if and only if there exist $\boldsymbol{u}', \boldsymbol{v}' \in (\Sigma^*)^n$ such that $\boldsymbol{u}' \preceq \boldsymbol{u}$, $\boldsymbol{v}' \in R_{wc}(\boldsymbol{u}')$, and $\boldsymbol{v} \preceq \boldsymbol{v}'$.

The definition of the mappings R_c and R_{wc} are extended straightforwardly to sets of vectors of words. We denote by R_c^* and R_{wc}^* the reflexive-transitive closure of R_c and R_{wc} respectively. Given a language L, $R_c^*(L)$ (resp. $R_{wc}^*(L)$ is called its *cyclic closure* (resp. *weak cyclic closure*) by R.

Cyclic rewriting systems can be used to model processes communicating through unbounded FIFO queues (channels). Intuitively, a rule $r : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$ corresponds to an action which, for a queue of index i, receives the sequence x_i from the queue, and then sends the sequence y_i to the queue. Weak cyclic rewriting systems model processes communicating through unbounded *lossy* channels (which may lose messages at any time).

Cyclic rewriting systems correspond to Post tag systems. It is well know that they are Turing machine powerful, and hence, the set $R_c^*(L)$ is in general not constructible, for any language L. On the other hand, we know by Theorem 1 that for every n-dim language L and every set of rewriting rules R, the set $R_{wc}^*(L)$ is SRE definable. However, this set is not necessarily effectively constructible.

2.4 Substitutions and $0L$-systems

A *substitution* is a mapping $\phi : \Sigma \to 2^{\Sigma^*}$ which associates with each symbol $a \in \Sigma$ a language over Σ (which may be empty).

A substitution is extended to words by taking $\phi(a_1 \cdots a_n) = \phi(a_1) \cdots \phi(a_n)$ and $\phi(\epsilon) = \epsilon$, and then generalized straightforwardly to languages. We denote by ϕ^i the substitution obtained by i compositions of ϕ. Then, let $\phi^{\geq n}(a)$ denote the set $\bigcup_{i \geq n} \phi^i(a)$, for any $n \geq 0$ and $a \in \Sigma$. We denote by ϕ^* (resp. ϕ^+) the reflexive-transitive (resp. transitive) closure of ϕ, i.e., $\phi^{\geq 0}(a)$ (resp. $\phi^{\geq 1}(a)$).

A substitution ϕ is *finite* if, for every $a \in \Sigma$, $\phi(a)$ is a finite language. A substitution is *downward closed* if it associates with each symbol a downward closed language, and it is *complete* if, for every $a \in \Sigma$, $\phi(a)$ is nonempty. A *weak substitution* is a downward closed complete substitution.

Given a set of 1-dim context-free rewriting rules R, we define a substitution φ_R such that, for every $a \in \Sigma$, $\varphi_R(a) = \{rhs(r) : r \in R \text{ and } lhs(r) = a\}$. We define also a substitution ϕ_R such that, for every $a \in \Sigma$, $\phi_R(a) = \{\epsilon\} \cup \varphi_R(a){\downarrow}$. Notice that, since R is finite, both φ_R and ϕ_R are finite substitutions. Moreover, ϕ_R is a weak substitution whereas φ_R is general not necessarily complete and not downward closed. Since φ_R is not complete, $\varphi_R(a){\downarrow}$ is in general different from $\phi_R(a)$ (since $\emptyset{\downarrow} = \emptyset$).

A $0L$*-system* (resp. *weak* $0L$*-system*) is a pair (R, φ_R) (resp. (R, ϕ_R)) where R is a finite set of context-free rewriting rules.

The following lemmas show the link between cyclic context-free rewriting and substitution (parallel rewriting).

Lemma 1. *For every $u, v \in \Sigma^*$, $v \in \phi_R(u)$ if and only if $v \in R_{wc}^{|u|}(u)$.*

Lemma 2. *For every finite set of 1-dim context-free rules R, and every language L, if $\phi_R^*(L)$ is an effective SRE set, then $R_{wc}^*(L)$ is also an effective SRE set.*

Proof. Using Lemma 1, it is easy to show that $R_{wc}^*(L) = \{v\phi_R(u) \ : \ uv \in \phi_R^*(L)\}$. The result follows from Theorem 1 and the fact that for every effectively regular set S and regular substitution ϕ, the set $\{v\phi(u) \ : \ uv \in S\}$ is effectively regular (regular languages are closed under conjugation and regular substitution). □

3 Constructibility for Context-Free Systems

We give in this section our main result (Theorem 2) and present its proof. This result says that the language generated by a weak cyclic context-free rewriting system is an effective regular (actually SRE) set.

An adaptation of the proof for this theorem can be done to establish a result concerning the languages generated by $0L$-systems. This result says (see Theorem 3) that the set of subwords of such a language is an effective regular (SRE) set.

Theorem 2. *For every context-free language L (effectively described by a context-free grammar), and for every finite set of (1-dim) context-free rewriting rules R, the set $R_{wc}^*(L)$ is effectively SRE representable.*

Proof. Clearly, $R_{wc}^*(L) = R_{wc}^*(L{\downarrow})$ for every language L. Hence, by Proposition 1 and Theorem 1, it is sufficient to prove that $R_{wc}^*(L)$ is an effective SRE set for every SRE language L. Moreover, by Lemma 2, this can be done by proving that $\phi_R^*(L)$ is an effective SRE set. For that, we need several definitions and notations.

Given a finite weak substitution ϕ (effectively given), we define $\Sigma_\phi(a) = \phi^*(a) \cap \Sigma$. Intuitively, $\Sigma_\phi(a)$ is the set of accessible symbols from a by iterated substitutions. Notice that the set $\Sigma_\phi(a)$ can be computed since the alphabet Σ is finite.

We partition the set of symbols Σ according to the following equivalence relation: two symbols a and b are equivalent if $\Sigma_\phi(a) = \Sigma_\phi(b)$. The equivalence class of a symbol a is denoted $[a]$.

The accessibility relation induces an ordering between equivalences classes: $[a] \leq [b]$ if and only if $\Sigma_\phi(a) \subseteq \Sigma_\phi(b)$. Let $level(a)$ be the level of $[a]$ according to this ordering: minimal elements (equivalence classes) have level 0, and level $k+1$ corresponds to minimal elements of the set of classes without the elements of level less than k.

Given a finite weak substitution ϕ, a symbol a is *recursive* if $a \in \phi^+(a)$. It is *expansive* if $aa \in \phi^+(a)$. A symbol is *linear* if it is recursive and not expansive. Let us denote by Rec_ϕ (resp. Exp_ϕ, Lin_ϕ, $NRec_\phi$) the set of recursive (resp. expansive, linear, nonrecursive) symbols according to ϕ. Deciding whether a symbol is recursive, linear, or expansive is straightforward.

Notice that for every $a, b \in \Sigma$ such that $[a] = [b]$, we have $a \in Rec_\phi$ (resp. Exp_ϕ, Lin_ϕ, $NRec_\phi$) iff $b \in Rec_\phi$ (resp. Exp_ϕ, Lin_ϕ, $NRec_\phi$).

A substitution ϕ is *normal* if, for every $a \in Exp_\phi$ (resp. Lin_ϕ), the set $\phi(a)$ contains aa (resp. a).

Let R be a finite set of 1-dim context-free rewriting rules. We suppose hereafter that R does not contain rules of the form $\epsilon \mapsto y$. The consideration of this kind of rules is not difficult.

The following lemma says that is order to compute the image by ϕ_R^* of a regular language, it is sufficient to know how to compute this image for each symbol in the alphabet.

Lemma 3. *For every effective regular language L, if for every $a \in \Sigma$ the set $\phi_R^*(a)$ is effective, then the set $\phi_R^*(L)$ is also effective.*

Proof. We construct a finite-state transducer A which substitutes to each read symbol the set $\phi_R^*(a)$. Then, the language $\phi_R^*(L)$ is the output of A when applied to L. $\square$

Now, for every $a \in Exp_{\phi_R}$ (resp. Lin_ϕ), let $\pi(a)$ be the smallest strictly positive integer such that aa (resp. a) is in $\phi_R^{\pi(a)}$. Then, let $\pi = \mathrm{lcm}\,\{\pi(a) \;:\; a \in Rec_{\phi_R}\}$, and let $\psi_R = \phi_R^\pi$. It is easy to see that ψ_R is a normal finite weak substitution.

Lemma 4. *For every finite set of 1-dim context-free rewriting rules R, and every language L, if the set $\psi_R^*(L)$ is effective, then the set $\phi_R^*(L)$ is effective.*

Proof. For every symbol $a \in \Sigma$, we have $\phi_R^*(L) = \bigcup_{i=1}^{\pi-1} \phi_R^i(\psi_R^*(L))$. $\square$

By Lemma 4, and since ψ_R^* is a normal finite weak substitution, in order to compute the image by ϕ_R^* of a language L, it is sufficient to know how to compute the image of L for every given normal finite weak substitution. Moreover, by Lemma 3, it is sufficient to know how to do this for each symbol of the alphabet. The fact that we can do this is stated by the following key proposition.

Proposition 2. *For every normal finite weak substitution ϕ, and every symbol $a \in \Sigma$, the set $\phi^*(a)$ is effectively SRE representable.*

Proof. To present the proof, we need to introduce some notions and to establish several lemmas.

Let ϕ be a substitution. Then, for any positive integer B, we say that a symbol a is B-regular if $\phi^{B+n}(a) \subseteq \phi^{B+n+1}(a)$ for every $n \geq 0$. This means that, after the B first applications of ϕ, the iteration of ϕ yields an increasing sequence of sets (i.e., each application of ϕ augments or at least preserves the given set).

Remark 1. By definition, if a symbol is B-regular, then it is necessarily B' regular for every $B' \geq B$.

The following lemma says that all recursive symbols, hence all linear and expansive symbols, are 0-regular.

Lemma 5. *Let ϕ be normal finite weak substitution. Then, for every $a \in Rec_\phi$, a is 0-regular.*

Proof. Since $a \in \phi(a)$. $\square$

The following lemma shows that $\phi^*(a)$ is effective in the easy case of an expansive symbol a.

Lemma 6. *Let ϕ be a normal finite weak substitution. Then, for every $a \in Exp_\phi$, $\phi^*(a) = (\Sigma_\phi(a))^*$.*

Proof. First, it is clear that $\phi^*(a) \subseteq (\Sigma_\phi(a))^*$ since all the words in $\phi^*(a)$ are over the alphabet $\Sigma_\phi(a)$. Consider the reverse inclusion. By definition, we know that $a \in \phi(a)$ and $aa \in \phi(a)$. This means that $a^* \subseteq \phi^*(a)$. Thus, we must have $\phi^*(a^*) \subseteq \phi^*(a)$. Now, since ϕ is a substitution, we have also $\phi^*(a^*) = (\phi^*(a))^*$. Hence, we obtain that $(\phi^*(a))^* \subseteq \phi^*(a)$. Finally, since $\Sigma_\phi(a) \subseteq \phi^*(a)$, we have $(\Sigma_\phi(a))^* \subseteq (\phi^*(a))^*$, from which we deduce that $(\Sigma_\phi(a))^* \subseteq \phi^*(a)$. □

Now, we give the main lemma which allows to prove Proposition 2. The proof of this lemma itself involves several lemmas (Lemmas 8, 9, 10, 11, 12, 13, and 14) which allow to establish that $\phi^*(a)$ is effective in the case where a is a linear symbol.

Lemma 7. *Let ϕ be a normal finite weak substitution. Then, for every $a \in \Sigma$, there exists a $B \geq 0$ such that a is B-regular and $\phi^*(a)$ is effectively regular.*

Proof. The proof is by induction on the level of the symbols.

Basis: Let $a \in \Sigma$ such that $level(a) = 0$. If a is nonrecursive, then $\phi(a) = \epsilon$, and hence, a is 1-regular, and $\phi^*(a) = \epsilon$ is effectively regular. If a is expansive, then by Lemma 5 and Lemma 6, we know that a is 0-regular and $\phi^*(a)$ is effectively regular. If a is linear, then by Lemma 5 a is 0-regular. Moreover, since $level(a) = 0$, we have $\phi(a) = \Sigma_\phi(a)$, and hence $\phi^*(a) = [a]$ is effectively regular.

Inductive step: Let us now consider a symbol a such that $level(a) = k + 1$ for some $k \geq 0$, and assume, by induction hypothesis and Remark 1, that there exists $B \geq 0$ such that every symbol b of level at most k is B-regular, and $\phi^*(b)$ is effectively regular. We show that the symbol a is $(B+1)$-regular and that $\phi^*(a)$ is effectively regular.

Case 1: a is nonrecursive. Then, for every $b \in \Sigma_\phi(a)$, $level(b) < level(a)$. By induction hypothesis, for every $b \in \Sigma_\phi(a)$, b is B-regular and $\phi^*(b)$ is effective. It easy to see that a is $(B + 1)$-regular and $\phi^*(a)$ is effectively regular.

Case 2: a is expansive. By Lemma 5, we know that a is 0-regular, and hence, a is $(B + 1)$-regular (for any $B \geq 0$). Moreover, by Lemma 6, we know that $\phi^*(a)$ is effectively regular.

Case 3: a is linear. We know by Lemma 5 that a is 0-regular, and hence that it is $(B + 1)$-regular. It remains to show that $\phi^*(a)$ is effectively regular.

Let us consider the context-free grammar $G_a = (N_a, T_a, S_a, \rho_a)$ such that:

- the set of nonterminal symbols is $N_a = [a]$,
- the set of terminal symbols is $T_a = \Sigma_\phi(a) \setminus [a]$,
- the start symbol is $S_a = a$,
- the set of rules is $\rho_a = \{b \to w \ : \ b \in [a] \text{ and } w \in \phi(b)\}$.

Notice that, since a is linear, the rules of G_a are either *nonterminal rules* of the form $b \to ub'v$, where $b' \in [a]$ and $u, v \in T_a^*$, or *terminal rules* of the form $b \to w$ where $w \in T_a^*$. Since ϕ is normal, we have by definition rules of the form $b \to b$ for any $b \in [a]$.

We index the nonterminal rule from 1 to n_1 and the terminal rules from 1 to n_2, where n_1 (resp. n_2) is the number of nonterminal (resp. terminal) rules in G_a. Then, we derive from G_a another context-free grammar $\widehat{G_a} = (N_a, \widehat{T_a}, S_a, \widehat{\rho_a})$ where:

– $\widehat{T_a} = \{\langle u\rangle_i, \langle v\rangle_i \ : \ \exists r = b \to ub'v \in \rho_a.\ index(r) = i\} \cup \{\langle\langle w\rangle\rangle_i \ : \ \exists r = b \to w \in \rho_a.\ w \in T_a^*$ and $index(r) = i\}$.
– $\widehat{\rho_a} = \{b \to \langle u\rangle_i b' \langle v\rangle_i \ : \ \exists r = b \to ub'v \in \rho.\ index(r) = i\} \cup \{b \to \langle\langle w\rangle\rangle_i \ : \ \exists r = b \to w \in \rho.\ w \in T_a^*$ and $index(r) = i\}$.

The following lemmas show the links between derivation in the grammar $\widehat{G_a}$ and the iterations of the substitution ϕ.

Lemma 8. *If there is a derivation in $\widehat{G_a}$ of length ℓ from a to some nonterminal word, then this word is of the form $\langle u_\ell\rangle_{i_\ell} \cdots \langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1} b \langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2} \cdots \langle v_\ell\rangle_{i_\ell}$, and*

$$\phi^{\ell-1}(u_\ell)\cdots\phi(u_2)u_1 b v_1 \phi(v_2)\cdots\phi^{\ell-1}(v_\ell) \subseteq \phi^\ell(a).$$

Lemma 9. *For every word $ubv \in \phi^\ell(a)$ where $b \in [a]$, there exists a derivation in $\widehat{G_a}$ of length ℓ from a to some nonterminal word $\langle u_\ell\rangle_{i_\ell} \cdots \langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1} b \langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2} \cdots \langle v_\ell\rangle_{i_\ell}$ such that $u \in \phi^{\ell-1}(u_\ell)\cdots\phi(u_2)u_1$, and $v \in v_1\phi(v_2)\cdots\phi^{\ell-1}(v_\ell)$.*

Lemma 10. *If there is a derivation in $\widehat{G_a}$ of length ℓ from a to some terminal word, then this word is of the form $\langle u_\ell\rangle_{i_\ell} \cdots \langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1} \langle\langle w\rangle\rangle_i \langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2} \cdots \langle v_\ell\rangle_{i_\ell}$, and*

$$\phi^{\ell}(u_\ell)\cdots\phi^2(u_2)\phi(u_1) w \phi(v_1)\phi^2(v_2)\cdots\phi^{\ell}(v_\ell) \subseteq \phi^\ell(a).$$

The proofs of the three lemmas above are by straightforward inductions.

Lemma 11. *For every terminal word σ in $\phi^\ell(a)$ (i.e., $\sigma \in T_a^* \cap \phi^\ell(a)$), there exists an integer $m \leq \ell$ and a derivation in $\widehat{G_a}$ of length m from a to some terminal word $\mu = \langle u_m\rangle_{i_m} \cdots \langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1} \langle\langle w\rangle\rangle_i \langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2} \cdots \langle v_m\rangle_{i_m}$ such that*

$$\sigma \in \phi^{\ell-m}[\phi^m(u_m)\cdots\phi^2(u_2)\phi(u_1) w \phi(v_1)\phi^2(v_2)\cdots\phi^m(v_m)].$$

Proof. The fact that $\sigma \in \phi^\ell(a)$ implies that there are words $\sigma_0, \sigma_1, \ldots, \sigma_\ell$ such that $\sigma_0 = a$, $\sigma = \sigma_\ell$, and $\sigma_{i+1} \in \phi(\sigma_i)$ for every $i \in \{0, \ldots, \ell - 1\}$. Let σ_i be the last word in this sequence which contains a symbol $b \in [a]$. This means that after i applications of ϕ, we get a terminal word σ_{i+1}. Let $m = i + 1$ and $\mu = \sigma_m$. By Lemma 9, we know that there exists a derivation in $\widehat{G_a}$ of length $i = m - 1$ from a to some nonterminal word $\mu' = \langle u_\ell\rangle_{i_\ell} \cdots \langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1} b \langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2} \cdots \langle v_\ell\rangle_{i_\ell}$ such that $\mu \in \phi^{m-1}(u_m)\cdots\phi(u_2)u_1 b v_1\phi(v_2)\cdots\phi^{m-1}(v_m)$. The result follows from the fact that μ can be derived in $\widehat{G_a}$ from μ' using a terminal rule, and from the fact that $\sigma \in \phi^{\ell-m}(\mu)$. □

Now, the following lemmas show the key property which allows to compute $\phi^*(a)$ by iterating the application of ϕ only a finite number of times, thanks to B-regularity.

Lemma 12. *If there is a derivation in $\widehat{G_a}$ of length ℓ from a to some nonterminal word $\langle u_\ell\rangle_{i_\ell} \cdots \langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1} b \langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2} \cdots \langle v_\ell\rangle_{i_\ell}$, then*

$$\begin{aligned}\phi^{\geq B}(u_\ell)\cdots\phi^{\geq B}(u_{B+1})\phi^{B-1}(u_B)\cdots\phi(u_2)u_1 b \\ v_1\phi(v_2)\cdots\phi^{B-1}(v_B)\phi^{\geq B}(v_{B+1})\cdots\phi^{\geq B}(v_\ell) \subseteq \phi^*(a)\end{aligned}$$

Proof. Suppose that there is a derivation δ in $\widehat{G_a}$ from a to the nonterminal word $\alpha = \langle u_\ell\rangle_{i_\ell}\cdots\langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1} b\langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2}\cdots\langle v_\ell\rangle_{i_\ell}$, and consider a word σ in the set

$$\begin{aligned}\phi^{\geq B}(u_\ell)\cdots\phi^{\geq B}(u_{B+1})\phi^{B-1}(u_B)\cdots\phi(u_2)u_1 b\\ v_1\phi(v_2)\cdots\phi^{B-1}(v_B)\phi^{\geq B}(v_{B+1})\cdots\phi^{\geq B}(v_\ell).\end{aligned} \tag{1}$$

Hence, σ can be written $\mu_\ell\cdots\mu_{B+1}\mu_B\cdots\mu_2\mu_1 b\nu_1\nu_2\cdots\nu_B\nu_{B+1}\cdots\nu_\ell$ where $\mu_i\in\phi^{i-1}(u_i)$ and $\nu_i\in\phi^{i-1}(v_i)$ for every $i\in\{1,\ldots,B\}$, and for every $i\in\{B+1,\ldots,\ell\}$, there is $m_i, n_i \geq B$ such that $\mu_i\in\phi^{m_i}(u_i)$ and $\nu_i\in\phi^{n_i}(v_i)$. Let $\kappa=\max\{n_i, m_i \,:\, B+1\leq i\leq\ell\}$. Then, using the property of B-regularity (see Remark 1), we deduce from (1) that σ is in the set:

$$\begin{aligned}\phi^{\kappa+\ell-(B+1)}(u_\ell)\cdots\phi^{\kappa}(u_{B+1})\phi^{B-1}(u_B)\cdots\phi(u_2)u_1 b\\ v_1\phi(v_2)\cdots\phi^{B-1}(v_B)\phi^{\kappa}(v_{B+1})\cdots\phi^{\kappa+\ell-(B+1)}(v_\ell).\end{aligned} \tag{2}$$

Now, consider the derivation δ in $\widehat{G_a}$ leading to the word α defined in the beginning of the proof. This derivation can be decomposed as $a \stackrel{*}{\Longrightarrow} \beta \stackrel{*}{\Longrightarrow} \alpha$, where $\beta = \langle u_\ell\rangle_{i_\ell}\cdots\langle u_{B+1}\rangle_{i_{B+1}} b'\langle v_{B+1}\rangle_{i_{B+1}}\cdots\langle v_\ell\rangle_{i_\ell}$ where $b'\in[a]$. This means that we have a sub-derivation δ' of the form $b' \stackrel{*}{\Longrightarrow} \langle u_B\rangle_{i_B}\cdots\langle u_1\rangle_{i_1} b\langle v_1\rangle_{i_1}\cdots\langle v_B\rangle_{i_B}$. We define another derivation in $\widehat{G_a}$ which produces first the word β as in δ, then iterates $\kappa - B$ times the rule $b'\to\langle\epsilon\rangle b'\langle\epsilon\rangle$ (recall that $b'\to b'$ must exist in G_a since the original substitution ϕ is normal), and finally applies the rules of δ'. This derivation produces a nonterminal word α' of the form

$$\begin{aligned}\langle u_\ell\rangle_{i_{\kappa+\ell-B}}\cdots\langle u_{B+1}\rangle_{i_{\kappa+1}}\langle\epsilon\rangle^{\kappa-B}\langle u_B\rangle_{i_B}\cdots\langle u_1\rangle_{i_1} b\\ \langle v_1\rangle_{i_1}\cdots\langle v_B\rangle_{i_B}\langle\epsilon\rangle^{\kappa-B}\langle v_{B+1}\rangle_{i_{\kappa+1}}\cdots\langle v_\ell\rangle_{i_{\kappa+\ell-B}}.\end{aligned}$$

Then, we deduce from (2) and Lemma 8 that $\sigma\in\phi^*(a)$. This terminates the proof of Lemma 12. □

Lemma 13. *If there is a derivation in $\widehat{G_a}$ of length ℓ from a to some terminal word $\langle u_\ell\rangle_{i_\ell}\cdots\langle u_2\rangle_{i_2}\langle u_1\rangle_{i_1}\langle\langle w\rangle\rangle_i\langle v_1\rangle_{i_1}\langle v_2\rangle_{i_2}\cdots\langle v_\ell\rangle_{i_\ell}$, then*

$$\phi^{\geq B}(u_\ell)\cdots\phi^{\geq B}(u_B)\phi^{B-1}(u_{B-1})\cdots\phi(u_1)w\phi(v_1)\cdots\phi^{B-1}(v_{B-1})\phi^{\geq B}(v_B)\cdots\phi^{\geq B}(v_\ell)\subseteq\phi^*(a).$$

Proof. Similar to the proof of Lemma 12. □

Now, we are able to give a construction for $\phi^*(a)$:

Consider the nonterminal words $\langle u_\ell\rangle_{i_\ell}\cdots\langle u_1\rangle_{i_1} b\langle v_1\rangle_{i_1}\cdots\langle v_\ell\rangle_{i_\ell}$ generated by the grammar $\widehat{G_a}$. They form an effective linear context-free language. By a regular transduction, we can decorate the terminal symbols in these words by indices in $\{0,1,...,B\}$ in the following manner: each nonterminal symbol $\langle u_j\rangle_{i_j}$ or $\langle v_j\rangle_{i_j}$ is decorated by $j-1$ if $j<B$, otherwise by B. This yields another effective context-free linear language. Now, by induction hypothesis, we know that for every symbol b in T_a, the set $\phi^*(b)$ is effective, and hence, the set $\phi^{\geq B}(u)$ is effective, for any $u\in T_a^*$. Therefore, we can use another transduction to substitute the finite set $\phi^j(u)$ to each nonterminal symbol $\langle u\rangle_i$ decorated by an index $j<B$, and to substitute the (effective) regular set $\phi^{\geq B}(u)$ to each $\langle u\rangle_i$ decorated by the index B. We get in this manner a new linear context-free language, we call L_1.

We apply a similar process to terminal words $\langle u_\ell\rangle_{i_\ell}\cdots\langle u_1\rangle_{i_1}\langle\langle w\rangle\rangle_i\langle v_1\rangle_{i_1}\cdots\langle v_\ell\rangle_{i_\ell}$ generated by $\widehat{G_a}$. In this case, the central symbol $\langle\langle w\rangle\rangle_i$ is decorated by index 0, and each nonterminal symbol $\langle u_j\rangle_{i_j}$ or $\langle v_j\rangle_{i_j}$ is decorated by j if $j < B$, otherwise by B. As in the previous cases, we obtain an effective linear language. Let us call this language L_2^0. Then, we repeat this process, but this time, we move up all the indices by one. In this manner, the regular substitution performed after the indexing step replaces $\langle\langle w\rangle\rangle_i$ by $\phi(w)$, all the $\langle u_j\rangle_{i_j}$'s such that $j < B-1$ by $\phi^{j+1}(u_j)$, and all the $\langle u_j\rangle_{i_j}$'s such that $j \geq B-1$ by $\phi^B(u_j)$. Let us call L_2^1 the so obtained linear language. We repeat this process B times. The union of all the languages L_2^j is an effective linear language L_2.

Lemma 14. *The language $L_1 \cup L_2$ is precisely $\phi^*(a)$ and it is effectively regular.*

Proof. By Lemma 12 and Lemma 13, we know that $L_1 \cup L_2 \subseteq \phi^*(a)$. Let us consider the reverse inclusion. Let σ be a word in $\phi^*(a)$. There are two cases to consider:

First, suppose that σ is a nonterminal word (i.e., of the form ubv where $b \in [a]$). Then, by Lemma 9 we know that $\sigma \in L_1$.

Consider now that σ is a terminal word (i.e. $\sigma \in T_a^*$). Then, by Lemma 11, there exists an intermediate terminal word σ' such that $\sigma' \in L_2^0$ and $\sigma \in \phi^n(\sigma)$ for some $n \geq 0$. It can be seen that, if $n < B$, then $\sigma \in L_2^n$, otherwise $\sigma \in L_2^B$. In both cases, $\sigma \in L_2$. This terminates the proof of the fact that $L_1 \cup L_2 = \phi^*(a)$.

As for the effective regularity of $L_1 \cup L_2$, we know that this language is an effective context-free language, but we know also that it must be a downward closed language since ϕ is downward closed. Then, the result follows from Proposition 1. ☐

Lemma 14 terminates the case where a is a linear symbol in the inductive step of the proof of Lemma 7. Hence, Lemma 7 is proved. ☐

Proposition 2 follows from Lemma 7 and Theorem 1. ☐

Theorem 2 follows from Lemma 2, Lemma 3, Lemma 4, and Proposition 2. ☐

Remark 2. By proving the result above (in fact from Lemma 3, Lemma 4, and Proposition 2), we have proved that the set of words generated by any weak $0L$-system is an SRE effective set.

Now, we can adapt the proof presented above in order to show that the set of subwords of the language generated by any $0L$-system is an effective SRE set.

Theorem 3. *For every context-free language L, and every $0L$-system (R, φ_R), the set $\varphi_R^*(L){\downarrow}$ is effectively SRE representable.*

Proof. (Hints:) We define the set of *mortal* symbols as the smallest set $M_R \subseteq \Sigma$ which includes the set of all symbols a such that $\phi(a) = \emptyset$, and such that, for every $a \in \Sigma$, if $\phi(a) \subseteq M_R^*$, then $a \in M_R$. Intuitively, mortal symbols are symbols for which the rewriting is eventually blocking, i.e., after a finite number of rewriting steps we get $\emptyset$. (Notice that mortal symbols are necessarily non-recursive). It is easy to see that, since we have a finite alphabet, the set M_R is computable in at most $|\Sigma|$ steps.

Then, consider the set of rules R' which is the restriction of R to non-mortal symbols (i.e., the set of all the rules that do not involve a mortal symbol). It can be seen that, for every $a \in \Sigma \setminus M_R$, $\varphi_{R'}^*(a){\downarrow} = \phi_{R'}^*(a)$. Moreover, the set $\phi_{R'}^*(a)$ can be computed by Proposition 2.

Now, to get $\varphi_R^*(a)\!\downarrow$, we have to take into account the mortal symbols and include all the words (and their subwords) these symbols can generate. For that, we compute $\varphi^{|\Sigma|}(\phi_{R'}^*(a))$, which is clearly an effective regular set, an take its downward closure. □

4 Unconstructibility results

We prove in this section that the weak cyclic closure is unconstructible in general, even though we know that it is always regular. For that, we use known results on *lossy channel systems*. Such a system consists of a finite-state machine operating on a single FIFO buffer which is unbounded and lossy in the sense that it can nondeterministically lose messages. Notice that FIFO-channel systems operate in general on several channels, but it is sufficient to consider the case of one channel to establish our results.

Formally, we assume a finite set M of *messages*. Then, a *Lossy Channel System (LCS)* over M is a pair $\mathcal{L} = (S, T)$, where S is a finite set of *control states* and T is a finite set of *transitions*. Each transition is a triple (s_1, op, s_2) where $s_1, s_2 \in S$ and op is a *channel operation*. A channel operation is either a *send* operation of the form $!m$ which appends the message m to the end of the channel, or a *receive operation* of the form $?m$ which removes the message m from the head of the channel (if m is not in the head of the channel then the receive transition is not enabled).

A *configuration* γ of $\mathcal{L}$ is a pair (s, w) where $s \in S$ and w is a string (word) over M. Intuitively, s defines the control state of the finite-state part, while w defines the contents of the channel. We define a transition relation $\Longrightarrow_{\mathcal{L}}$ on configurations such that $(s_1, w_1) \Longrightarrow_{\mathcal{L}} (s_2, w_2)$ if and only if one of the following conditions is satisfied:

- $w_2 = w_1 m$ and $(s_1, !m, s_2) \in T$.
- $w_1 = m w_2$ and $(s_1, ?m, s_2) \in T$.
- $w_1 = w m w''$, $w_2 = w w''$, and $s_1 = s_2$. This corresponds to losing message m in the channel. Notice that the control state is not changed.

We use $\stackrel{*}{\Longrightarrow}_{\mathcal{L}}$ to denote the reflexive-transitive closure of $\Longrightarrow_{\mathcal{L}}$. For a configuration γ, we let $\mathit{Reach}_{\mathcal{L}}(\gamma)$ be the set $\{\gamma' \;:\; \gamma \stackrel{*}{\Longrightarrow}_{\mathcal{L}} \gamma'\}$.

Given any LCS $\mathcal{L}$, we can deduce from Theorem 1 that $\mathit{Reach}_{\mathcal{L}}(\gamma)$ is SRE definable. However, from [May00] we know the following:

Lemma 15. *For each configuration γ, the set $\mathit{Reach}_{\mathcal{L}}(\gamma)$ is in general not computable.*

We use this lemma to establish that the weak cyclic closure is not constructible already for 1-dim rewriting systems. We show that this holds actually as soon as we allow rules with two symbols in their left hand side.

Theorem 4. *There is no algorithm which can construct the set $R_{wc}^*(\epsilon)$ for any given finite set of 1-dim rewriting rules R.*

Proof. Given a lossy channel system $\mathcal{L} = (S, T)$ over a set of messages M, we derive a 1-dim cyclic rewriting system $\mathcal{R}$ which "simulates" $\mathcal{L}$. We define $\Sigma = S \cup M$, and define $\mathcal{R} = (R, R_c)$ where R is the the smallest set of rules such that:

- *send rules:* for each $(s_1, !m, s_2) \in T$, there is $s_1 \mapsto m s_2 \in R$.
- *receive rules:* for each $(s_1, ?m, s_2) \in T$, there is $s_1 m \mapsto s_2 \in R$.
- *rotation rule:* for each $a \in \Sigma$, there is $a \mapsto a \in R$.

Intuitively each configuration (s, w) in $\mathcal{L}$ is encoded by a string $(w''sw')$ where $w = w'w''$. A send (resp. receive) transition is simulated by first using the rotation rules until we obtain a control state in the head of the string, and then rewriting with the corresponding send (resp. receive) rule. Lossiness is covered by the semantics of the weak rewriting relation.

We prove the following lemma:

Lemma 16. *Consider two configurations $\gamma_1 = (s_1, w_1)$ and $\gamma_2 = (s_2, w_2)$, and strings w_3 and w_4 with $w_1 = w_3w_4$. Then $\gamma_1 \stackrel{*}{\Longrightarrow}_{\mathcal{L}} \gamma_2$ if and only if there are w_5 and w_6 such that $w_2 = w_5w_6$ and $w_6s_2w_5 \in R^*_{wc}(w_4s_1w_3)$.*

Proof. **1.** First, we show that if $(s_1, w_1) \Longrightarrow_{\mathcal{L}} (s_2, w_2)$ and $w_1 = w_3w_4$, then there are w_5 and w_6 such that $w_2 = w_5w_6$ and $w_6s_2w_5 \in R^*_{wc}(w_4s_1w_3)$. There are three cases corresponding to the three types of transitions available in lossy channel systems:

1. If $w_2 = w_1m$ and $(s_1, !m, s_2) \in T$ for some $m \in M$. By the rotation rules $s_1w_3w_4 \in R^*_{wc}(w_4s_1w_3)$. By the send rules $w_3w_4ms_2 \in R_{wc}(s_1w_3w_4)$. We define $w_5 = \epsilon$ and $w_6 = w_3w_4m$.

2. If $w_1 = mw_2$ and $(s_1, ?m, s_2) \in T$ for some $m \in M$.

If $w_3 = \epsilon$ then we know that $w_4 = mw'_4$ for some w'_4. By the rotation rules $s_1mw'_4 \in R^*_{wc}(mw'_4s_1)$. By the receive rules $w'_4s_2 \in R_{wc}(s_1mw'_4)$. We define $w_5 = \epsilon$ and $w_6 = w'_4$.

If $w_3 \neq \epsilon$ then we know that $w_3 = mw'_3$ for some w'_3. By the rotation rules $s_1mw'_3w_4 \in R^*_{wc}(w_4s_1mw'_3)$. By the receive rules $w'_3w_4s_2 \in R_{wc}(s_1mw'_3w_4)$. We define $w_5 = \epsilon$ and $w_6 = w'_3w_4$.

3. If $w_1 = w'mw''$, $w_2 = w'w''$, and $s_1 = s_2$.

If $w'm$ is a prefix of w_3, i.e. if there is a w_7 such that $w_3 = w'mw_7$ and $w'' = w_7w_4$. By the rotation rules $mw_7w_4s_1w' \in R^*_{wc}(w_4s_1w'mw_7)$. Due to lossiness, we have $w_7w_4s_1w' \in R_{wc}(mw_7w_4s_1w')$. We define $w_5 = w'$ and $w_6 = w_7w_4$.

If w_3 is a prefix of $w'm$ then the proof is similar to above.

2. Now, we show that if $w_6s_2w_5 \in R_{wc}(w_4s_1w_3)$, then $(s_1, w_3w_4) \stackrel{*}{\Longrightarrow}_{\mathcal{L}} (s_2, w_5w_6)$. There are three cases corresponding to the three possible types of rules.

1. *send rules*: if $w_4 = \epsilon$, $w_6 = w_3m$, $w_5 = \epsilon$, and there is a transition (s_1, m, s_2). The resut follows immediately from the definition of $\Longrightarrow_{\mathcal{L}}$.

2. *receive rules*: similar to the case with send rules.

3. *rotation rules*: In this case $w_3w_4 = w_5w_6$ and $s_1 = s_2$. The result follows immediately. □

Corollary 1. *For configurations $\gamma_1 = (s_1, w_1)$ and $\gamma_2 = (s_2, w_2)$, we have $\gamma_1 \stackrel{*}{\Longrightarrow}_{\mathcal{L}} \gamma_2$ if and only if $s_2w_2 \in R^*_{wc}(s_1w_1)$.*

Proof. The *if* direction follows immediately from Lemma 16. For the *only if* direction, suppose that $\gamma_1 \stackrel{*}{\Longrightarrow}_{\mathcal{L}} \gamma_2$. Then, by Lemma 16 we know that there are w_5 and w_6 such that $w_2 = w_5w_6$ and $w_6s_2w_5 \in R^*_{wc}(s_1w_1)$. Using the rotation rules we get $s_2w_5w_6 \in R^*_{wc}(w_6s_2w_5)$. This ends the proof of Corollary 1. □

Theorem 4 follows directly from Lemma 15 and Corollary 1. □

We show now that in the case of context-free rewriting systems, the weak cyclic closure is in general not constructible starting from dimension 2.

Theorem 5. *There is no algorithm which can construct the set $R^*_{wc}(\epsilon, \epsilon)$ for any given finite set of 2-dim context-free rewriting rules R.*

Proof. The actions of a LCS can be straightforwardly encoded as rules of a 2-dim context-free cyclic rewriting system: one of the two dimensions can be used to encode the control state, and the other one to encode the lossy channel. □

5 Constructibility for n-dim Context-Free Rings

We show in this section that the weak cyclic closure is constructible for a special kind of 2-dim context-free rewriting systems.

A set of n-dim rewriting rules R is a *ring* if, for every rule $r : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$ in R, $\exists i \in \{1, \ldots, n\}$. $\forall j \neq i.\ x_j = \epsilon$ and $\forall j \neq (i+1) \bmod n.\ y_j = \epsilon$. This means that r is either of the form $(\epsilon, \ldots, \epsilon, x_i, \epsilon, \ldots, \epsilon) \mapsto (\epsilon, \ldots, \epsilon, y_{i+1}, \epsilon, \ldots, \epsilon)$ or of the form $(\epsilon, \ldots, \epsilon, x_n) \mapsto (y_1, \epsilon, \ldots, \epsilon)$. Notice that every set of 1-dim rewriting rules is a 1-dim ring and vice-versa.

We consider hereafter n-dim context-free rings, i.e., the x_i's are symbols in Σ. Intuitively, the rules of these systems correspond to actions of FIFO-channel systems where a symbol x is received from some channel of index i, and a word y is sent to the channel of index $(i+1) \bmod n$.

Theorem 6. *For every n-dim context-free ring R, and every n-dim language L which is a product of n context-free languages (effectively described by context-free grammars), the set $R^*_{wc}(L)$ is effectively SRE representable.*

Proof. (Hints:) We give here the main idea of the proof. Let us consider the 2-dim case. We suppose that we have disjoint alphabets Σ_1 and Σ_2 for each of the two dimensions. The set of rules can be partitioned into two sets: the set of $(1,2)$-rules of the form $r_1 : (a, \epsilon) \mapsto (\epsilon, \beta)$, and the set of $(2,1)$-rules of the form $r_2 : (\epsilon, b) \mapsto (\alpha, \epsilon)$. Consider a vector $(au, bv) \in \Sigma_1^* \times \Sigma_2^*$, a $(1,2)$-rule r_1, and a $(2,1)$-rule r_2. Then, it can be seen that the application of r_1 does not affect the applicability of r_2 and vice-versa: It is possible to apply either r_1 followed by r_2 or r_2 followed by r_1. In both cases, we get the same vector $(u\alpha, v\beta)$. From this observation, we can restrict our attention, in a first step, to the set of vectors that are reachable by a special kind of rewriting sequences, called *edge rewriting sequences*, which consists in alternating two phases: $(1,2)$-rules are first applied as much as possible (until queue 1 becomes empty), and then $(2,1)$-rules are applied as much as possible (until queue 2 becomes empty), and so on. Then, in a second step we construct the set of vectors that are reachable by all other (non-edge) rewriting sequences using (left and right) conjugation operations combined with substitutions.

Thus, the main problem is to construct the set of reachable vectors by edge rewriting sequences. For this, we proceed as follows: We construct from the original 2-dim ring R a 1-dim context-free system R' over the alphabet $\Sigma_1 \cup \Sigma_2$. For each $(1,2)$-rule of the form $(a, \epsilon) \mapsto (\epsilon, \beta)$ (resp. $(2,1)$-rule of the form $(\epsilon, b) \mapsto (\alpha, \epsilon)$) we define a rule $a \mapsto \beta$ (resp. $b \mapsto \alpha$). Then, we use Proposition 2 to compute the set of words $\phi_{R'}(L)$. This set is actually the union of two disjoint sets L_1 and L_2, where L_1 (resp. L_2) is a set of words u on Σ_1 (resp. Σ_2) corresponding to all reachable vectors of the form (u, ϵ) (resp. (ϵ, u)). □

References

[AAB99] Parosh Aziz Abdulla, Aurore Annichini, and Ahmed Bouajjani. Algorithmic verification of lossy channel systems: An application to the bounded retransmission protocol. In *Proc. TACAS '99, 5^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579 of *Lecture Notes in Computer Science*, 1999.

[ABJ98] Parosh Aziz Abdulla, Ahmed Bouajjani, and Bengt Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels. In *Proc. 10^{th} Int. Conf. on Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318, 1998.

[AJ96] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[BBK93] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the ACM*, (40):653–682, 1993.

[BG96] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In Alur and Henzinger, editors, *Proc. 8^{th} Int. Conf. on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, 1996.

[BGWW97] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *Proc. of the Fourth International Static Analysis Symposium*, Lecture Notes in Computer Science. Springer Verlag, 1997.

[BH97] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of fifo-channel systems with nonregular sets of configurations. In *Proc. ICALP '97*, volume 1256 of *Lecture Notes in Computer Science*, 1997.

[Bur98] O. Burkart. Queues as processes. *Electrnic Notes in Theoretical Computer Science*, 18, 1998.

[CFI96] Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable Channels Are Easier to Verify Than Perfect Channels. *Information and Computation*, 124(1):20–31, 1996.

[CHS92] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. In W. R. Cleaveland, editor, *Proc. CONCUR '92, Theories of Concurrency: Unification and Extension*, pages 138–147, 1992.

[Cou91] B. Courcelle. On constructing obstruction sets of words. *EATCS*, 44:178–185, June 1991.

[CVWY90] C. Courcoubetis, M. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. In *Proc. Workshop on Computer Aided Verification*, 1990.

[GH94] J.F. Groote and H. Hüttel. Undecidable equivelences for basic process algebra. *Information and Computation*, 1994.

[GvdP93] J.F. Groote and J. van de Pol. A bounded retransmission protocol for large data packets. Technical report, Department of Philosophy, Utrecht University, October 1993.

[Hol91] G.J. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall, 1991.

[May00] R. Mayr. Undecidable problems in unreliable computations. In *Theoretical Informatics (LATIN'2000)*, number 1776 in Lecture Notes in Computer Science, 2000.

[RS80] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L-systems.* Academic Press, 1980.

[Sti96] C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. In *Proc. CONCUR '96, 7^{th} Int. Conf. on Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 217–232. Springer Verlag, 1996.