A New Rewrite Method for Proving Convergence of Self-Stabilizing Systems

Joffroy Beauquier[1], Béatrice Bérard[2] and Laurent Fribourg[2]

[1] LRI, CNRS URA 410, Université Paris-Sud 91405 Orsay cedex, France
jb@lri.fr
[2] LSV, CNRS UMR 8643, ENS de Cachan, 61 av. du Prés. Wilson,
94235 Cachan cedex, France
{berard,fribourg}@lsv.ens-cachan.fr

Abstract. In the framework of self-stabilizing systems, the convergence proof is generally done by exhibiting a measure that strictly decreases until a legitimate configuration is reached. The discovery of such a measure is very specific and requires a deep understanding of the studied transition system. In contrast we propose here a simple method for proving convergence, which regards self-stabilizing systems as string rewrite systems, and adapts a procedure initially designed by Dershowitz for proving termination of string rewrite systems.

1 Introduction

Introduced by Dijkstra, with three mutual exclusion algorithms on a ring of processes [10], the notion of self-stabilization has been largely studied for the last ten years (see [28,30] for surveys). In this paper, we consider a system which consists of a ring of machines controlled by a "central demon". Its configuration is the concatenation of the components local states and it is characterized by a set R of transitions defined over configurations. The system is *self-stabilizing* with respect to a subset L of *legitimate* configurations when, regardless of the initial configuration and regardless of the transition selected each time by the central demon, it is guaranteed to arrive in a configuration of L within a finite number of steps. The set L is assumed to have a *closure* property: from a legitimate configuration in L, the system persistently remains in L. It is also frequent to assume that there is *no-deadlock*. With these two hypotheses, it is easy to show that a system is self-stabilizing iff it has the *no-cycle* property: there is no cyclic sequence of transitions which contains some configuration $w \notin L$. This property is often shown by exhibiting a *norm function* defined over the set of configurations, whose value strictly decreases after each transition (or each bounded sequence of transitions) as long as the configuration is not legitimate [30]. Since such a measure is usually very specific to the considered system, finding one is very difficult and requires a deep understanding of this system (see e.g.[22,13,4]).

We propose here a new approach for proving the absence of cycle. Configurations are viewed as words of a formal language, transitions of R as rewrite rules, and the no-cycle property as a variant of the nontermination property for rewrite rules. More precisely, we will attack the no-cycle property under the following equivalent form: there is no infinite sequence of transitions of R^{-1} (the reverse system obtained from R by switching origin and target configurations) starting from a configuration $w \notin L$. The absence of such infinite sequences will then be shown by refining the generation procedure of *reduction chains*[1], first proposed by Dershowitz for proving string rewriting termination [7]. The method proposed here is new in that: 1) it uses a general technique of string-rewriting to deal with self-stabilization; 2) instead of working with the direct rewriting relation from arbitrary configurations towards the legitimate ones, it uses the inverse relation; 3) it does not consider all the possible rewrite derivations, but only "representative" ones, using a restricted rewriting strategy.

Related work on self-stabilization proofs. [10] is without proof. In [11], a correctness proof is given for the third (3-state) algorithm of [10], by showing properties of executions using behavioral reasoning. As already pointed out, almost all further proof methods (cf. [30]) are based on norm functions (see an example in [18]) but, as expressed by Gouda [14]: "It has been my experience that the ratio of time to design a stabilizing system to the time to verify its stabilization is about one to ten". For simplifying proof process, general paradigms have been proposed: various protocol compositions making proofs modular ([3,15]), attractor or staircase methods ([14,28]), automatic transformations into stabilizing systems [2]. Some ideas in [17] are also used in our approach, as discussed in Section 7. Note that, recently, some works were done for proving convergence without appealing to a norm function: [31] uses techniques borrowed from control theory and [1] induction techniques over the set of configurations.

Related work on rewrite techniques applied to distributed systems. Although viewing transitions as rewrite rules is rather natural, the application of general rewrite techniques for proving properties of distributed systems has not been explored to our knowledge, except in [24–26] where graph rewriting techniques (with priorities) are used to prove the correctness of various distributed algorithms (election, spanning-tree construction,...). However this work does not address the issue of self-stabilization.

Plan of the paper. Section 2 gives an intuitive presentation of the method with the example of Ghosh's 4-state algorithm. Section 3 shows how self-stabilization can be viewed as a property of string rewriting systems. Our basic method and the underlying correctness result are explained in Section 4, then refined in Section 5. Section 6 relates compositionality results in the field of self-stabilization and in the field of rewriting. Section 7 concludes with final remarks and perspectives.

2 An illustrating example: Ghosh's 4-state algorithm

Ghosh's algorithm is a variant of Dijkstra's 4-state self-stabilizing algorithm [13]. The system consists of a parametric number N of machines $(0, 1, \cdots, N-1)$, which have four states: $\{0, 1, 2, 3\}$, except the *top* machine $N-1$ (resp. *bottom* machine 0) which has only two states: $\{0, 2\}$ (resp. $\{1, 3\}$). The configuration of the system is the string of all machine states, delimited by special end symbols '#'. Writing X, Y for nonempty string variables, the transitions correspond to the following system $R = Middle \cup Top \cup Bottom$ of rewrite rules.

- $Middle$ is made of rules of the form:

$M_1: \quad \#X(q+1)qY\# \rightarrow \#X(q+1)(q+1)Y\#$

[1] also called *forward closures* in [8].

$M_2: \quad \#Xq(q+1)Y\# \rightarrow \#X(q+1)(q+1)Y\#$
where $q \in \{0,1,2,3\}$ and '+' is addition modulo 4.

- *Top* is made of rules of the form:
  $T_1: \quad \#X32\# \rightarrow \#X30\#$ and $T_2: \quad \#X10\# \rightarrow \#X12\#$
- *Bottom* is made of rules of the form:
  $B_1: \quad \#12X\# \rightarrow \#32X\#$ and $B_2: \quad \#30X\# \rightarrow \#10X\#$

Ghosh proves the convergence by considering a norm function (Br, Ds) such that either Br or Ds *strictly* decreases each time a transition is done. Individually Br and Ds are non-increasing functions. Br is the number of *breaks*, i.e. the number of neighbouring states q, q' of the string which differ by at least one unit. Ds is a very subtle function measuring the sum of distances between pairs of neighbouring breaks of the string. All the difficulty of Ghosh's proof comes from the discovery of such a measure Ds. In contrast, we now informally explain how our method proves the convergence, with the help of measure Br only. Our first idea is to consider not the original system R but the inverse R^{-1} obtained by switching lefthand and righthand sides. Since Br is non-increasing via R, it is non-decreasing via R^{-1}. Our second idea is to prove convergence by showing that there is no infinite derivation via R^{-1} (except those among legitimate configurations, which are not displayed here for lack of space). We will focus on Br-preserving infinite derivations, i.e. infinite derivations which preserve the number of breaks: since Br is non-decreasing and bounded by the total number N of system components, any infinite derivation has an infinite Br-preserving suffix, so there is no loss of generality. For instance, the two following reductions (with underlines indicating positions of a substring to be reduced) are discarded because they increase Br by one:

$\#\underline{11}23(0123)\cdots(0123)2\cdots2\# \quad \rightarrow \quad \#1023(0123)\cdots(0123)2\cdots2\#$
$\#1123(0123)\cdots(0123)2\cdots\underline{222}\cdots2\# \rightarrow \quad \#1123(0123)\cdots(0123)2\cdots212\cdots2\#$

Our third and main idea is to generate not all the possible Br-preserving derivations via R^{-1} but only "representative" ones, in the sense that an infinite derivation exists iff an infinite representative one exists. As explained later, representative derivations are obtained by reasoning with 1st-order variables that represent arbitrary sets of strings, and instantiating these variables in a "minimal" way through successive steps. For pedagogical reasons, we first assume that appropriate instantiations are known from the beginning, and present the treatment of Ghosh's algorithm, starting from a string s_0 with all its variables already instantiated.

Generation of representative strings.

We start from:

$s_0: \quad \#\pi(0123)\cdots(0123)3\cdots\underline{30}\#$, where π is either 3 or 123. Now s_0 reduces via T_1^{-1} to
$s_1: \quad \#\pi(0123)\cdots(0123)3\cdots\underline{33}2\#$, which reduces via a (possibly empty) sequence of M_1^{-1} to
$s_2: \quad \#\pi(0123)\cdots(0123)3\cdots\underline{33}2\cdots2\#$.

At some point, one generates

$s_3: \quad \#\pi(0123)\cdots(0123)(012\underline{3)3}2\cdots2\#$, which reduces via M_2^{-1} successively to
$s_4: \quad \#\pi(0123)\cdots(0123)(01\underline{22}3)2\cdots2\#$,
$s_5: \quad \#\pi(0123)\cdots(0123)(0\underline{11}23)2\cdots2\#$,
$s_6: \quad \#\pi(0123)\cdots(0123)(\underline{00}123)2\cdots2\#$,
$s_7: \quad \#\pi(0123)\cdots(012\underline{3)(3}0123)2\cdots2\#$

A generalized form for s_3 and s_7 is

$s_8: \quad \#\pi(0123)\cdots(0123)(012\underline{33})(0123)\cdots(0123)2\cdots2\#$,

which contains a substring (01233) between a left and right sequences of (0123)'s.
s_8 rewrites via M_2^{-1} successively to

$s_9: \quad \#\pi(0123)\cdots(0123)(01\underline{22}3)(0123)\cdots(0123)2\cdots2\#$,
$s_{10}: \quad \#\pi(0123)\cdots(0123)(0\underline{11}23)(0123)\cdots(0123)2\cdots2\#$,

$s_{11}: \quad \#\pi(0123)\cdots(0123)(\underline{00}123)(0123)\cdots(0123)2\cdots2\#,$

$s_{12}: \quad \#\pi(0123)\cdots(012\underline{3)(3}0123)(0123)\cdots(0123)2\cdots2\#.$

which can be seen as a copy of s_8 where an element (0123) of the left sequence has moved to the right sequence. Such an iterated application of M_2^{-1} always ends when the left sequence is empty, thus yielding:

$s_{end}: \quad \#\pi(30123)(0123)\cdots(0123)2\cdots2\#.$

In case π is 3 there is no possible Br-preserving rule application.

In case π is 123, there are two ultimate applications of M_2^{-1} yielding successively:

$s'_{end}: \quad \#1\underline{22}3(0123)(0123)\cdots(0123)2\cdots2\#,$

$s''_{end}: \quad \#1123(0123)(0123)\cdots(0123)2\cdots2\#,$

with no Br-preserving rule applicable to s''_{end}.

This result shows that there is no infinite Br-preserving derivation starting from s_0. It remains to explain how the initial string s_0 was inferred. The basic idea is to reason on strings with variables at the first-order level. Variables are instantiated in a "minimal" manner so that reduction rules become applicable. We start with the left-hand side $\#X30\#$ of top rule T_1^{-1}, which is reduced to $\#X32\#$. Then X is bound to X_13, and instance $\#X_1332\#$ reduces via M_1^{-1} to $\#X_1322\#$. This operation is iterated, yielding $\#X_n32^n2\#$. Then X_n is instantiated with $Y3$, but now M_2^{-1} is applied, which yields $\#Y232^n2\#$. Then Y is instantiated with $Z2$, and application of M_2^{-1} yields $Z1232^n2\#$. Iterating this operation generates strings of the form $\#W_1(0123)2^n2\#, \cdots,$ $\#W_m(0123)^m2^n2\#, \cdots$. Similarly, another representative string derivation can be obtained, starting from the other top rule righthand side $\#X12\#$.

As a recapitulation, our strategy consists in generating "representative" chains of derivations by starting from righthand sides of top rules and applying Br-preserving rules (from right to left). One then shows that only finite derivations are obtained (except over legitimate configurations). The finiteness of derivations will be proved formally by finding a generic set Π of "growing patterns", i.e. a set of strings such that any Br-preserving rule applied to any element $t \in \Pi$ yields an element $u \in \Pi$ greater than t, for some ordering $>$ over Π. In the case of Ghosh's algorithm, Π is of the form: $\{\#X\pi'(0123)^m2^n2\# \mid m \geq 0, n \geq 0\}$ where π' is 3, 23, 123 or 0123. The associated relation $>$ merely orders elements of Π according to the number m of substrings (0123) they contain. (m is increased by one, each time four rules are applied consecutively.) Our underlying claim is that such a strictly monotonic ordering $>$ is much simpler to find than Ghosh's strictly monotonic norm (Br, Ds) because we focus on "representative" derivations using a restricted strategy of rule application, instead of considering all the possible configurations. The rest of the paper is devoted to the formal description of our strategy and a proof of its correctness.

3 Self-stabilizing systems as string rewrite systems

We first recall some basic definitions from (string) rewrite systems ([9],[5]). The words considered here are generally delimited by a leftmost and rightmost special symbol '#'. The symbols appearing between them belong to a finite alphabet Σ or a set $\mathcal{V} = \{W, X, Y \ldots\}$ of variable symbols. A *string* is an element of Σ^*, with ε for the empty string. A *ground word* is an element of $\#\Sigma^*\#$ and an *(open) word* is an element of $\#(\Sigma \cup \mathcal{V})^*\#$. A substitution is a mapping θ from $\mathcal{V}$ to $(\Sigma \cup \mathcal{V})^*$ with $\theta(W) = W$ almost everywhere except on a finite set of variables denoted $Dom(\theta)$. A substitution θ is represented by a finite set of pairs of the form $\{W/\theta(W)\}_{W\in Dom(\theta)}$. A substitution θ is *ground* when $\theta(W)$ is in Σ^*, for all $W \in Dom(\theta)$.

String rewrite systems. The string rewrite systems considered here contain length-preserving rules, divided into three subsets: *top rules* in $Tops_S$ are applied to the rightmost part of words; *bottom rules* in $Bottoms_S$ are applied to the leftmost part of words (or simultaneously at both ends); the rest of rules in $Middles_S$ are called *middle rules.* More precisely, let ℓ, r (resp. ℓ_i, r_i for $i = 1, 2$) be nonempty strings of Σ^* of the same length, and X, Y variables,

- $Middles_S$ is made of rules of the form: $\#X\ell Y\# \to \#XrY\#$
- $Tops_S$ is made of rules of the form: $\#X\ell\# \to \#Xr\#$
- $Bottoms_S$ is made of rules of the form:

$\#\ell X\# \to \#rX\#$ or $\#\ell_1 X\ell_2\# \to \#r_1Xr_2\#$.

We are going to apply these rules either to ground words or to words of the form $\#uWv\#$ where u, v denote strings over Σ^*, and W a variable.

Example.[2]
B_1^{-1} : $\#1X2\# \to \#2X1\#$ is a bottom-rule. T_4^{-1} : $\#X21\# \to \#X12\#$ is a top-rule. M_1^{-1} : $\#X01Y\# \to \#X10Y\#$ and M_4^{-1} : $\#X20Y\# \to \#X02Y\#$ are middle rules.

Ground reduction. A ground word w is *reducible* via a rule of the form $\#X\ell_1Y\# \to \#Xr_1Y\#$ iff $w = \#u\ell_1v\#$ for some strings $u, v \in \Sigma^*$. One also says that w is an instance of the rule lefthand side via the ground substitution $\{X/u, Y/v\}$. The *reduced form* of w is $w' = \#ur_1v\#$. Reduction via a rule of the form $\#\ell_1X\ell_2\# \to \#r_1Xr_2\#$ is defined in a similar way.

A word w reduces to w' via S, written $w \to_S w'$ (or sometimes simply $w \to w'$), if w' is the reduced form of w via some rule of S. We say that S is *non terminating* iff there exists an infinite sequence of reductions via S starting from some ground word w. Otherwise, S is said to be terminating.

Example. The word w : $\#1012\#$ reduces via rule B_1^{-1} : $\#1X2\# \to \#2X1\#$ to w' : $\#2011\#$, using the substitution $\{X/01\}$.

Self-stabilization. We are now able to give a formal definition of self-stabilization for a system modeled as a string rewrite system R. From now on, configurations are regarded as ground words. Writing L_N for the set of legitimate configurations in a system with N machines, we define the *global set of legitimitate configurations*, as $\mathcal{L} = \cup_{N\geq 2} L_N$.

Definition 1. A rewrite system R is *self-stabilizing* with respect to set $\mathcal{L}$ iff:

(0) Each ground word is reducible via R.

(1) $\mathcal{L}$ is *closed via* R, i.e: $w \in \mathcal{L} \wedge w \to_R w' \Rightarrow w' \in \mathcal{L}$, for all ground words w, w'.

(2) There is no ground cyclic derivation of the form $w_1 \to_R \cdots \to_R w_n = w_1$ with $w_1 \notin \mathcal{L}$.

[2] these rules are the inverse of B_1, T_4, M_1, M_4 from Beauquier-Debas algorithm (see section 5).

Statement (0) expresses a *no-deadlock* property, (1) a *closure* property for $\mathcal{L}$, and (2) a *no-cycle* property. It easily follows from this definition that any "maximal" derivation is infinite and reaches the set $\mathcal{L}$: this corresponds to a *convergence* property (also called *no-livelock* property in [6]).

Assuming (1), we have two equivalent versions for (2):

(2′) There is no ground cyclic derivation via $S \equiv R^{-1}$, of the form $w_1 \to_S \cdots \to_S w_n = w_1$ with $w_1 \notin \mathcal{L}$.

(2″) There is no infinite ground derivation via $S \equiv R^{-1}$ starting from a word $w \notin \mathcal{L}$.

Proof. (2) $\Leftrightarrow$ (2′) $\Leftrightarrow$ (2″). One has clearly (2) $\Leftrightarrow$ (2′) and (2″) $\Rightarrow$ (2′). Let us show (2′) $\Rightarrow$ (2″) by showing $\neg$(2″) $\Rightarrow$ $\neg$(2′). Suppose $\neg$(2″): there is an infinite ground derivation Δ via S: $u_1 \to_S \cdots \to_S u_i \to_S \cdots$ with $u_1 \notin \mathcal{L}$. Since S is length-preserving there should be a cycle in this derivation, so there is an initial part of Δ which is of the form: $u_1 \to_S \cdots \to_S u_j \to_S \cdots \to_S u_n = u_j$, for some j and n. Since $u_1 \to_S^* u_j$, we have $u_j \to_R^* u_1$. It follows $u_j \notin \mathcal{L}$ because, otherwise, u_1 would be itself in $\mathcal{L}$ by assumption (1) of closure. Therefore the subpart $u_j \to_S \cdots \to_S u_n = u_j$ is a cycle with $u_j \notin \mathcal{L}$. One has thus exhibited a cycle via S containing an element not in $\mathcal{L}$, which proves $\neg$(2′).

Remark 2. Note that equivalence between (2) and (2″) does not hold any longer if R is used instead of R^{-1} within (2″).

4 A first-order characterization of cycles

Assuming now that R satisfies (0) and (1), we will focus on the problem of proving the *no-cycle* property, stated under form (2″). Our method relies on a first-order characterization of cycles: we will show that an infinite ground derivation via R^{-1} (as mentioned in (2″)) is actually an instance of an infinite derivation at the "first-order" level. In order to state our main result, we need the notion of top chains, which is transposed from [7] in our particular context.

Minimal reductions and top chains. We now deal with one-variable words t of the form $\#uWv\#$, with $u, v \in \Sigma^*$ and $W \in \mathcal{V}$, and we consider the minimal complete set of substitutions $\mu_1, ..., \mu_k$ which make t reducible via a given rule of the form $\ell \to r$, i.e, the set of most general unifiers of t with a rule lefthand side ℓ. The general unification problem for words is complex, and was solved by Makanin [27]. However our particular unification problem here is simple, because t and ℓ do not share variables, and are "linear" (i.e. contain at most one occurrence of the same variable). In such a case the number of most general unifiers is finite: roughly speaking, it suffices to consider all the manners in which t and ℓ overlap depending on the possible instantiations of their variables (see, e.g., [20]). Suppose now that t and ℓ are unifiable via a set of most general unifiers $\mu_1, ..., \mu_k$ ($t\mu_i = \ell\mu_i$ for $i = 1, ..., k$), so that each instance $t\mu_i$ of t reduces to $r\mu_i$. We will disregard unifiers μ_j which instantiate t at a "variable position". We then say that t is *minimally reducible at a nonvariable position* via the set of unifiers $\{\mu_1, ..., \mu_k\}$. (This corresponds to the operation called "narrowing" in first-order rewrite systems [29,12,19].) The corresponding set of minimal reduction steps is $\{t\mu_i \to r\mu_i\}_{i=1,...,k}$.

Example.
Suppose we have a rule of the form $\#X11Y\# \to \#X22Y\#$. The word $t : \#1W1\#$ unifies with lefthand side $\#X11Y\#$ via most general unifiers:
$\mu_1 : \{W/1W'\} \cup \{X/\varepsilon, Y/W'1\}$, $\mu_2 : \{W/W'1\} \cup \{X/1W', Y/\varepsilon\}$,
$\mu_3 : \{W/\varepsilon\} \cup \{X/\varepsilon, Y/\varepsilon\}$ and $\mu_4 : \{W/W_1 11 W_2\} \cup \{X/1W_1, Y/W_2 1\}$.
The last unifier μ_4 (with the associated reduction) is discarded because it corresponds to a unification taking place at a variable position of t. The minimal reductions of t corresponding to μ_1, μ_2, μ_3 are: $\#11W'1\# \to \#22W'1\#$, $\#1W'11\# \to \#1W'22\#$ and $\#11\# \to \#22\#$.

Definition 3. The *top minimal reduction chains* (or simply *top chains*) of rewrite system S form a set of derivations inductively defined as follows:

- Every top rule $\#X\ell_1\# \to \#Xr_1\#$ of S is a top minimal reduction chain.
- If $C : \ v \to \cdots \to w$ is a top minimal reduction chain and $\ell \to r$ is a rule of S such that w unifies with ℓ at a nonvariable position via the set of most general unifiers $\{\mu_1, ..., \mu_k\}$, then $v\mu_i \to \cdots \to w\mu_i \to r\mu_i$ (for $i = 1, ..., k$) is a top minimal reduction chain, called *successor* of C via $\ell \to r$.

The transitive closure of the successor relation is called "iterated successor". It is easy to see that, due to the form of the rules of the string rewrite system S, a top chain $t_1 \to t_2 \to ... \to t_n$ is such that either all the words $t_1, ..., t_n$ are ground, or each t_i $(1 \leq i \leq n)$ is of the form $\#u_i W v_i\#$ where the u_is (resp. v_is) are strings of Σ^* of the same length.

Example. Consider the top rule T_4^{-1} : $\#W21\# \to \#W12\#$ viewed as a top chain. Its righthandside $\#W12\#$ unifies with the lefthand side of B_1^{-1} : $\#1X2\# \to \#2X1\#$ via the set of most general unifiers $\mu_1 : \{W/\varepsilon\} \cup \{X/\varepsilon\}$ and $\mu_2 : \{W/1W'\} \cup \{X/X'1\}$. Therefore the successors of T_4^{-1} via B_1^{-1} are:
$\#21\# \to \#12\# \to \#21\#$ and $\#1W'21\# \to \#1W'12\# \to \#2W'11\#$.

Note that the requirement that the starting chain is a top rule instead of a general rule of S is new with respect to Dershowitz's definition [7]. Such a requirement is made possible because we will assume additionally that system $S - Top_S$ is terminating (see Theorem 5).

Definition 4. A top chain (resp. ground derivation) $t_1 \to \cdots \to \cdots \to t_n$ is *quasi-cyclic* if $t_i = t_n$ for some $i < n$, and $t_p \neq t_q$ for all distinct p, q less than n.

A characterization of self-stabilization. We can now state our main result:

Theorem 5. *Let $R = Middle_R \cup Top_R \cup Bottom_R$ be a rewrite system and let $\mathcal{L}$ be a set of configurations such that:*
(0) each ground word is reducible via R,
(1) $\mathcal{L}$ is closed via R, and
(3) $R - Top_R$ (or, equivalently, $(R - Top_R)^{-1}$) is terminating.
Then R is self-stabilizing w.r.t. $\mathcal{L}$ iff there is no quasi-cyclic top chain $t_1 \to \cdots \to t_n$ via R^{-1}, such that $t'_1 \notin \mathcal{L}$ for some ground instance t'_1 of t_1.

The proof of theorem 5 involves additional notions of "active" or "inactive" steps within infinite ground derivations, as introduced by Dershowitz [7]. The corresponding definitions and properties, as well as the proof, are given in the Appendix.

In order to mechanically check point (3) of theorem 5, i.e. termination of $R - Top_R$, one can use classical well-founded orderings used in rewriting theory [9]. One can also use Dershowitz's chain test: generate all the general chains until either one "cycles" in the sense of [7] (non-termination detection) or all terminate (termination proof). Dershowitz's procedure can also be refined if one knows that $R - Top_R - Bottom_R$ is itself terminating. Then, in order to prove termination of $R - Top_R$, it suffices to generate only the "bottom" chains via $R - Top_R$, i.e. chains that start from a bottom rule, and check that none of them has a cycle. (Of course, in this case, one has to check additionally the termination of $R - Top_R - Bottom_R$, but this is generally easier.)

5 A practical proof method

We now present our basic procedure, as well as a fully treated example of application.

5.1 Basic procedure and ϕ-refinement

Theorem 5 suggests to prove self-stabilization by generating all top chains via R^{-1} and checking the condition over the initial word t_1 for quasi-cyclic chains. Unfortunately, top chains, when generated in a brute manner, are frequently in infinite number. However it is often possible to discover some recurrent forms, called "patterns", for words t_n appearing at the end of chains. For example, starting from word $\#2W11\#$, and applying Beauquier-Debas system $S \equiv R'^{-1}$ (see below) with rules $\#X20Y\# \rightarrow \#X02Y\#$ and $\#X01Y\# \rightarrow \#X10Y\#$ (with $Y \neq \varepsilon$), one generates chains that all end with words of the form $\#0^j2W10^k1\#$ with $j, k \geq 0$. Formally:

Definition 6. A *set of patterns* Π is a set of the form
$\{\ \#a_1^{i_1}...a_p^{i_p}Wb_1^{j_1}...b_q^{j_q}\#\ \mid\ \psi(i_1,...,i_p,j_1,...,j_q)\ \}$
where $a_1,...,a_p,b_1,...,b_q$ are letters of Σ, $i_1,...,i_p,j_1,...,j_q$ are natural numbers, and ψ an arithmetical relation.

The discovery of such patterns allows us to characterize finitely an infinite number of top chains. These patterns will be required to be *closed* via S, i.e.: for all $\pi \in \Pi$ and all minimal reduction of π to π' via S using unifier μ ($\pi\mu \rightarrow \pi'$), π' is in Π. Additionally, π' will be required to be greater than $\pi\mu$ for some ordering $>$ over words, *compatible* with substitution i.e. such that: $t_1 > t_2 \Longrightarrow t_1\theta > t_2\theta$, for any ground substitution θ. Such requirements prevent cycles within derivations of the form $t_1 \rightarrow \cdots \rightarrow t_n \rightarrow t_{n+1} \rightarrow ... \rightarrow t_{n+m}$, as far as $t_n \in \Pi$ and no word of Π unifies with t_i $(1 \leq i < n)$. More precisely:

Definition and Property 7. An *S-closed set of growing patterns* Π is a set of patterns such that, for all $\pi \in \Pi$ and all minimal reduction of π to π' via S using μ ($\pi\mu \rightarrow \pi'$): $\pi' \in \Pi$ and $\pi' > \pi\mu$ for some ordering $>$ compatible with substitution.
If a non quasi-cyclic chain $C:\ t_1 \rightarrow \cdots \rightarrow t_n$ is such that t_n belongs to an S-closed set Π of growing patterns nonunifiable with any $t_1,...,t_{n-1}$, then any iterated successor of C of the form: $t_1(\mu_1\cdots\mu_m) \rightarrow \cdots \rightarrow t_n(\mu_1\cdots\mu_m) \rightarrow t_{n+1}(\mu_1\cdots\mu_m) \rightarrow \cdots \rightarrow t_{n+m}(\mu_1\cdots\mu_m)$ is itself non quasi-cyclic.

Example. Consider the set Π : $\{\#0^j2W10^k1\# \mid j \geq 0, k \geq 0\}$. The only rules of

Beauquier-Debas system $S \equiv R'^{-1}$ (see section 5.2) that are applicable to Π are: $\#X20Y\# \rightarrow \#X02Y\#$ and $\#X01Y\# \rightarrow \#X10Y\#$, with $Y \neq \varepsilon$.
Given π : $\#0^j2W10^k1\#$, the first rule (resp. second rule) minimally reduces it to π'_1 : $\#0^{j+1}2W10^k1\#$ (resp. π'_2 : $\#0^j2W10^{k+1}1\#$), which belongs to Π and is greater than π with respect to the number of '0's to the left of '2' (resp. number of '0's to the right of leftmost occurrence of '1'). Therefore Π is an S-closed set of growing patterns.

Using theorem 5 and property 7, and assuming $(0), (1), (3)$ for R, we obtain:

Basic Procedure
Start with set of top rules $Tops_S$ (with $S \equiv R^{-1}$) as an initial set of chains.
For each chain $t_1 \rightarrow \cdots \rightarrow t_n$, compute iteratively its successors via rules of S unless:
- $t_1 \in \mathcal{L}$, or
- $t_n \in \Pi$, for some S-closed set Π of growing patterns nonunifiable with any $t_1, ..., t_{n-1}$.

If, among all the generated chains, there is one quasi-cyclic chain $t_1 \rightarrow \cdots \rightarrow t_i \rightarrow \cdots \rightarrow t_n = t_i$ such that t_1 has a ground instance $t'_1 \notin \mathcal{L}$, then R is not self-stabilizing. Otherwise R is self-stabilizing.

Remark 7. In order to detect if, given t_1, there exists a ground instance not belonging to $\mathcal{L}$, one may assume that $\mathcal{L}$ is characterized by a *regular language* (see [17]). If t_1 is ground, then the problem reduces to $t_1 \notin \mathcal{L}$, which is decidable. If t_1 is not ground, then t_1 is of the form $\#uWv\#$, and the problem reduces to test $\#u\Sigma^*v\# \cap \overline{\mathcal{L}} \neq \emptyset$, which is also decidable. Note besides that the assumption of regularity for $\mathcal{L}$ allows us to mechanically check property (1) of closure via R. This is equivalent to check that the image of $\mathcal{L}$ via $\rightarrow_R$ is included into $\mathcal{L}$. Since $\mathcal{L}$ is regular and $\rightarrow_R$ can be seen as a rational transduction, this inclusion problem is decidable (see [23],[17]).

Our basic procedure can be refined by using a measure, say ϕ, over ground words which "never increases" when applying a rule, such as Br in section 2. Finding such a measure is generally much easier than finding a norm, which must "always decrease" when applying a rule (or a bounded number of rules) of R. Formally, we assume given a measure ϕ from words over $(\Sigma \cup \mathcal{V})^*$ to $\mathbb{N}$, with the following properties:
- ϕ is *non-increasing with* R, i.e., is such that $t \rightarrow_R t'$ implies $\phi(t) \geq \phi(t')$.
- ϕ is *compatible with substitution*: $\phi(t) > \phi(t')$ implies $\phi(t\theta) > \phi(t'\theta)$ for any ground substitution θ.

Given ϕ, it is easy to show that the basic procedure can be refined by generating only ϕ*-preserving* successors via $S \equiv R^{-1}$ instead of all the possible successors. (A successor $t_1\mu \rightarrow \cdots \rightarrow t_n\mu \rightarrow t_{n+1}\mu$ of $t_1 \rightarrow \cdots \rightarrow t_n$ is ϕ-preserving iff $\phi(t_n\mu) = \phi(t_{n+1}\mu)$.)

5.2 Beauquier-Debas algorithm

This system originates from [4], and is an adaptation of Dijkstra's third (3-state) algorithm [10]. In our formalism, it corresponds to the following system R:

Bottom	B_1 :	$\#2X1\# \rightarrow \#1X2\#$
Top	T_1 :	$\#X00\# \rightarrow \#X21\#$
	T_2 :	$\#X10\# \rightarrow \#X01\#$
	T_3 :	$\#X20\# \rightarrow \#X11\#$

	T_4 :	$\#X12\# \to \#X21\#$	
	T_5 :	$\#X22\# \to \#X01\#$	
Middle	M_1 :	$\#X10Y\# \to \#X01Y\#$	(with $Y \neq \varepsilon$)[3]
	M_2 :	$\#X11Y\# \to \#X02Y\#$	(with $Y \neq \varepsilon$)
	M_3 :	$\#X12Y\# \to \#X00Y\#$	(with $Y \neq \varepsilon$)
	M_4 :	$\#X02Y\# \to \#X20Y\#$	(with $Y \neq \varepsilon$)
	M_5 :	$\#X22Y\# \to \#X10Y\#$	(with $Y \neq \varepsilon$)

$\mathcal{L}$ is defined as: $\#0^*20^*1\# \ \cup\ \#0^*10^*2\#$.

In this example, it is assumed that the sum of the elements of the initial configuration is null, modulo 3. This property is preserved when applying the rules of R. One checks easily that any ground word (with a null sum of elements) is reducible via R, and $\mathcal{L}$ is closed via R (see [4]). Therefore, R is self-stabilizing iff there is no ground cyclic derivation via R containing an element $w \notin \mathcal{L}$. As remarked in [4], one can see that T_1, T_2, T_3 are applied at most once. As a consequence R is self-stabilizing iff there is no ground cyclic derivation via $R_0 \equiv R - \{T_1, T_2, T_3\}$ containing an element $w \notin \mathcal{L}$. The measure ϕ over a word $t \in (\Sigma \cup \mathcal{V})^*$, is defined as the number of nonnull elements contained by t. Obviously, ϕ is non-increasing with R_0 and compatible with substitution. Besides, among rules of R_0 only rules $B_1^{-1}, M_1^{-1}, M_4^{-1}, T_4^{-1}$ preserve the number of nonnull elements. The ϕ-refinement of the basic procedure thus consists in generating top chains via $R'^{-1} \equiv \{B_1^{-1}, M_1^{-1}, M_4^{-1}, T_4^{-1}\}$ instead of R_0^{-1}. The chains generated this way are all derived from the single initial top chain (corresponding to T_4^{-1}):

(c_0) $\quad \#X21\# \to \#X12\#.$

Immediate successors of (c_0) are obtained by minimally reducing word $\#X12\#$ either via M_1^{-1} or via B_1^{-1}. In the first case, we generate:

(c_0^1) $\quad \#X021\# \to \#X012\# \to \#X102\#.$

In the second case, we generate:

(c_{11}) $\quad \#21\# \to \#12\# \to \#21\#.$

(c_{12}) $\quad \#1X21\# \to \#1X12\# \to \#2X11\#.$

More generally, the first successors of (c_0) are obtained by applying i times $(i \geq 0)$ rule M_1^{-1}, then possibly rule B_1^{-1}. The application of rule M_1^{-1} i times to (c_0) yields:

(c_0^i) $\quad \#X0^i21\# \to \#X0^i12\# \to \#X0^{i-1}102\# \to \cdots \to \#X10^i2\#.$

The application of B_1^{-1} then yields:

(c_{11}^i) $\quad \#0^i21\# \to \#0^i12\# \to \#0^{i-1}102\# \to \cdots \to \#10^i2\# \to \#20^i1\#.$

(c_{12}^i) $\quad \#1X0^i21\# \to \#1X0^i12\# \to \#1X0^{i-1}102\# \to \cdots \to \#1X10^i2\# \to \#2X10^i1\#.$

The first element $\#0^i21\#$ of (c_{11}^i) belongs to $\mathcal{L}$, so there is no need to compute the successors of (c_{11}^i). The last element $\#2X10^i1\#$ of (c_{12}^i) belongs to $\Pi : \{\#0^j2X10^k1\# \mid j \geq 0, k \geq 0\}$, which is an R'^{-1}-closed set of growing patterns (see example above). Furthermore, the other elements of (c_{12}^i): $\#1X0^i21\#, \#1X0^i12\#, \cdots, \#1X10^i2\#$ cannot unify with any element of Π (as they start with 1 instead of 0 or 2). Therefore we do not need to compute the successors of (c_{12}^i) either. So the procedure of top chain generation ends. The only quasi-cyclic chain generated is $(c_{11}^0) \equiv (c_{11}) \equiv \#21\# \to \#12\# \to \#21\#$, which starts with an element of $\mathcal{L}$. Self-stabilization is thus proved for Beauquier-Debas's variant of Dijkstra's 3-state algorithm. Self-stabilization of Ghosh's algorithm [13] can be proved formally along the same lines, as sketched out in section 2.

[3] Such a rule M_1 with condition $Y \neq \varepsilon$ should be regarded as an abbreviation for 3 rules M_{1a} : $\#X10aY\# \to \#X01aY\#$, with $a \in \{0, 1, 2\}$.

6 Compositionality

It is interesting to relate compositionality results obtained in the context of self-stabilizing systems with those obtained in the context of rewrite systems. Similarly to what Dershowitz proved w.r.t. composition of terminating systems (see theorem in [7], p.456), one can derive from theorem 5 sufficient conditions for self-stabilization of the combination of two self-stabilizing systems.

Theorem 8. *Assume that:*
- R_1 is self-stabilizing w.r.t. $\mathcal{L}_1$.
- R_2 is self-stabilizing w.r.t. $\mathcal{L}_2$ when starting from $\mathcal{L}_1$. [4]
- there is no overlap between lefthand sides of R_1 and righthand sides of R_2.
- all executions are fair w.r.t. both R_1 and R_2. [5]
Then $R_1 \cup R_2$ is self-stabilizing w.r.t. $\mathcal{L}_2$.

This can be seen as a version Herman's compositionality result [16] in our context.

7 Conclusion and perspectives

In contrast with methods relying on the existence of a strictly decreasing norm function [4,13,18,22], our method requires only little specific knowledge and proposes a uniform framework for the full proof of several non trivial examples, as shown here on Ghosh's 4-state algorithm [13] and Beauquier-Debas's 3-state algorithm [4]. These examples are simple ones, which allow us to give a clear view of the procedure. Besides, the corresponding problem is interesting in itself because it is related to mutual exclusion and has a non trivial specification. Our procedure is inspired by Dershowitz's chain generation procedure [7], and proves convergence of self-stabilizing algorithms much in the same way as Dershowitz proves the termination of rewrite systems. The method is not fully automatic: we need in particular to infer by hand generic "patterns" of configurations from words produced recurrently throughout derivations. Note that patterns have been also used in previous works on self-stabilization, e.g. in [17] where they are expressed under the form of regular languages. The main differences with [17] (and other traditional proof methods of self-stabilization) come from:

1. considering the *reverse* transition relation R^{-1} instead of R;
2. focusing on derivations (via R^{-1}) originating from *top*- configurations instead of derivations (via R) starting from any configuration.
3. deriving new configurations (via R^{-1}) through a restricted strategy of rule application (top chain generation) instead of deriving all the possible successor configurations (via R).

We have also given a natural counterpart of Herman's compositionality result in our framework. We are currently investigating an extension of our proof method in two directions. We first want to consider more realistic token ring algorithms, like different versions of the IBM token ring or FDDI protocols, involving a change from the state reading model to the message passing model. In this case, channel states must be modeled

[4] "when starting from $\mathcal{L}_1$" means that condition (0) is replaced with "(0′) Each ground word of $\mathcal{L}_1$ is reducible via R_2" in definition 1 of self-stabilization for R_2 w.r.t. $\mathcal{L}_2$.

[5] See, e.g., [30], p. 476, for a formal definition of "fairness".

and the hypotheses must be slightly modified to consider self-stabilization. The second natural way is an extension to algorithms running on arbitrary (non-ring) networks. In this framework, strings must be replaced by graphs. We would then have to use graph rewriting techniques, as proposed in [24–26] for other properties of distributed systems. As a promising first step, we consider the case of tree networks, in which leaves can receive markers, in the same way that top and bottom nodes receive '#'.

Acknowledgement

We would like to thank Nachum Dershowitz for his encouragement and pointing out reference [31].

References

1. G. Antonoiu and P.K. Srimani. "A Self-Stabilizing Leader Election Algorithm for Tree Graphs", *Journal of Parallel and Distributed Computing, 34:2*, May 1996, pp. 227–232.
2. Y. Afek, S. Kutten and M. Yung M.. "Memory efficient self stabilizing protocols for general networks", *Proc. 7th WDAG*, LNCS 725, Springer-Verlag, 1990, pp. 15-28.
3. A. Arora, M.G. Gouda and T. Herman. "Composite routing protocols". *Proc. 2nd IEEE Symp. on Parallel and Distributed Processing*, 1990.
4. J. Beauquier and O. Debas. "An optimal self-stabilizing algorithm for mutual exclusion on uniform bidirectional rings". *Proc. 2nd Workshop on Self-Stabilizing Systems*, Las Vegas, 1995, pp. 226-239.
5. R.V. Book and F. Otto. *String-Rewriting Systems*. Springer-Verlag, 1993.
6. J.E. Burns and J. Pachl. "Uniform Self-Stabilizing Rings". *TOPLAS* 11:2, 1989.
7. N. Dershowitz. "Termination of Linear Rewriting Systems". *Proc. ICALP*, LNCS 115, Springer-Verlag, 1981, pp. 448-458.
8. N. Dershowitz and C. Hoot. "Topics in termination". *Proc. Rewriting Techniques and Applications*, LNCS 690, Springer-Verlag, 1993, pp. 198-212.
9. N. Dershowitz and J.-P. Jouannaud. "Rewrite Systems". *Handbook of Theoretical Computer Science*, vol. B, Elsevier - MIT Press, 1990, pp. 243-320.
10. E.W. Dijkstra. "Self-stabilizing systems in spite of distributed control". *Comm. ACM* 17:11, 1974, pp. 643-644.
11. E.W. Dijkstra. "A Belated Proof of Self-stabilization". *Distributed Computing* 1:1, 1986, pp. 5-6.
12. M. Fay. "First-order unification in an equational theory". *Proc. 4th Workshop on Automated Deduction*, Austin, Texas, 1979, pp. 161-167.
13. S. Ghosh. "An Alternative Solution to a Problem on Self-Stabilization". *ACM TOPLAS* 15:4, 1993, pp. 735-742.
14. M.G. Gouda. "The triumph and tribulation of system stabilization". *Proc. 9th WDAG*, LNCS 972, Springer-Verlag, 1995, pp. 1-18.
15. M.G. Gouda and T. Herman. "Adaptative programming". *IEEE Transactions on Software Engineering* 17, 1991, pp.911-921.
16. T. Herman. *Adaptativity through distributed convergence*. Ph.D. Thesis, University of Texas at Austin. 1991.
17. J.-H. Hoepman. "Uniform Deterministic Self-Stabilizing Ring-Orientation on Odd-Length Rings". *Proc. 8th Workshop on Distributed Algorithms*, LNCS 857, Springer-Verlag. 1994, pp.265-279.
18. S.C. Hsu and S.T. Huang. "A self-stabilizing algorithm for maximal matching". *Information Processing Letters* 43:2, 1992, pp. 77-81.

19. J.-M. Hullot. "Canonical Forms and Unification". *Proc. 5th Conf. on Automated Deduction*, LNCS, Springer-Verlag, 1980, pp. 318-334.
20. J. Jaffar. "Minimal and Complete Word Unification". *J. ACM* 37:1, 1990, pp. 47-85.
21. S. Katz and K.J. Perry. "Self-stabilizing extensions for message-passing systems". *Distributed Computing* 7, 1993, pp. 17-26.
22. J.L.W. Kessels. "An Exercise in proving self-stabilization with a variant function". *Information and Processing Letters* 29, 1988, pp. 39-42.
23. Y. Kesten, O. Maler, M. Marcuse, A. Pnueli and E. Shahar. "Symbolic Model-Checking with Rich Assertional Languages". *Proc. CAV'97*, LNCS 1254, Springer-Verlag, 1997, pp. 424-435.
24. I. Litovski and Y. Métivier. "Computing with Graph Rewriting Systems with Priorities". *Theoretical Computer Science* 115:2, 1993, pp. 191-224.
25. I. Litovski, Y. Métivier and E. Sopena. "Different Local Controls for Graph Relabeling Systems". *Mathematical Systems Theory* 28:1, 1995, pp. 41-65.
26. I. Litovski, Y. Métivier and W. Zielonka. "On the Recognition of Families of Graphs with Local Computations". *Information and Computation* 118:1, 1995, pp. 110-119.
27. G.S. Makanin. "The problem of solvability of equations in a free semigroup". *Matematiceskij Sbornik* 103, 1977, pp. 147-236.
28. M. Schneider. "Self-Stabilization". *ACM Computing Surveys* 25:1, 1993.
29. J.R. Slagle. "Automated Theorem-Proving for Theories with Simplifiers, Commutativity, and Associativity." *J. ACM* 21:4, 1974, pp. 622-642.
30. G. Tel. *Introduction to Distributed Algorithms.* Cambridge University Press, 1994.
31. O. Theel and F.C. Gärtner. "An Exercise in Proving Convergence through Transfer Functions". *Proc. 4th Workshop on Self-stabilizing Systems*, Austin, Texas, 1999, pp. 41-47.

Appendix: proof of theorem 5

The proof of theorem 5 relies on properties involving the notions of active or inactive steps for derivations, as defined by Dershowitz [7].

Definition 9. The *active area* of a ground word w_i in a ground derivation $w_1 \to w_2 \to \cdots \to w_n$ is that part of w_i that has been created by the nonvariable portions of the righthand sides of the rules that have been applied. Only the top letter (i.e., the rightmost letter) of the initial word w_1 is considered active.

More precisely, suppose that a rule of the form $\#X\ell_1Y\# \to \#Xr_1Y\#$ (resp. $\#\ell_1X\ell_2\# \to \#r_1Xr_2\#$) is applied to a ground word w of the form $\#u_1\ell_1u_2\#$ (resp. $\#\ell_1u_1\ell_2\#$) to obtain a ground word w' of the form $\#u_1r_1u_2\#$ (resp. $\#r_1u_1r_2\#$). Then, in w', all the letters of r_1 (resp. r_1 and r_2) are active if at least one letter of ℓ_1 (resp. ℓ_1 or ℓ_2) was active; besides, all the letters of u_1, u_2 that were already active in w remain active in w'. We say that a ground word is *active* if its top letter is (active letters will be overlined).

Definition 10. An *active ground derivation* via S (resp. *inactive ground derivation* via S) is a ground derivation $w_1 \to w_2 \to \cdots \to w_n$ in which rules of S are applied only in the active area (resp. inactive area) of words.
We will denote by $\xrightarrow{act}$ (resp. $\xrightarrow{inact}$) an application of a rule at an active area (resp. inactive area) of a ground word.

We now give a property A.1, which is the counterpart of the relation between reduction sequences and narrowing sequences in first-order terms theory [19], and a property A.2, which was proved by Dershowitz [7].

Property A.1 (lifting lemma).
For all active ground derivation $\Delta : w_1 \xrightarrow{act} w_2 \xrightarrow{act} \cdots \xrightarrow{act} w_n$ via S, there exists a top chain $C : t_1 \to t_2 \to \cdots \to t_n$ via S which has Δ as an instance (i.e. such that $w_1 = t_1\theta, w_2 = t_2\theta, ..., w_n = t_n\theta$, for some ground substitution θ).

Property A.2 (semi-commutation).
Let w_1, w_2, w_3 be ground words such that $w_1 \xrightarrow{act} w_2 \xrightarrow{inact} w_3$. Then there exists a ground word w_2' such that $w_1 \xrightarrow{inact} w_2' \xrightarrow{act} w_3$.

We finally give two properties A.3 and A.4, which are adapted from [7] in our context.

Property A.3. Let S be a rewrite sytem such that $S - Top_S$ is terminating. Then any infinite ground derivation via S has only a finite number of inactive steps.

Proof. Consider an infinite ground derivation $\Delta :\ w_1 \to \cdots \to w_n \to \cdots$ via S. Applying a rule at an active area of a ground word cannot create any new inactive letters, while applying a rule at an inactive area only replaces a certain portion of inactive area by another inactive portion of the same length. Therefore either (a) all the inactive subareas of Δ disappear after a finite number of steps, or (b) at least one of them remains, but between two fixed positions. In case (b), there is a subpart Δ' of Δ of the form $w_i \to \cdots \to w_n \to \cdots$, such that every w_n is of the form $x_n v_n y_n$, all the x_n's (resp. v_n's, w_n's) have the same length, and the v_n's always remain inactive. Since the active application of rules over subparts of x_n or y_n have no influence over the inactive portion v_n, one can extract from Δ' an infinite inactive ground derivation Δ'' affecting only the v_n's. This infinite derivation Δ'' never makes use of a top rule since the top letter is active. This is in contradiction with assumption that $S - Top_S$ is terminating. The only possible case is therefore (a): all the inactive subareas disappear after a finite number of steps.

Property A.4. Suppose that $S - Top_S$ is terminating, and $\mathcal{L}$ is closed via S^{-1}. Then there is an infinite ground derivation via S starting from a word $w \notin \mathcal{L}$ if and only if there is a quasi-cyclic top chain via S, starting from a word t such that $t\theta \notin \mathcal{L}$ for some ground substitution θ.

Proof. The if part is obvious. To prove the only-if part, consider an infinite ground derivation Δ starting from a word $w \notin \mathcal{L}$. By property A.3 (since $S - Top_S$ is terminating) this infinite derivation contains only a finite number of inactive steps. Applying iteratively property A.2, one can push back these inactive steps to the beginning of the derivation, thus obtaining a reordered infinite ground derivation Δ'. Frome some point on, there are only active steps in derivation Δ'. Let Δ'' : $w_i \to w_{i+1} \to \cdots$ denote this active infinite ground part of derivation. Since $\mathcal{L}$ is closed via S^{-1}, we must have $w_i \notin \mathcal{L}$. Besides, since the rules are length-preserving, there is an initial part of Δ'' of the form $w_i \to \cdots \to w_j \to \cdots \to w_n$ such that $w_j = w_n$ and $w_p \neq w_q$ for all $p < q < n$. By the lifting lemma, there is a top chain $C : t_i \to \cdots \to t_j \to \cdots \to t_n$ with $C\theta = \Delta$, for some ground substitution θ. In particular $t_j\theta = w_j = w_n = t_n\theta$. But t_j and t_n are either both ground or of the form $\#u_jWv_j\#$ and $\#u_nWv_n\#$ with $|u_j| = |u_n|$ and $|v_j| = |v_n|$. So $t_j = t_n$ follows from $t_j\theta = t_n\theta$. Besides, for all $p < q < n$, t_p and t_q are distinct since their instances w_p, w_q via θ are distinct. Therefore $t_i \to \cdots \to t_j \to \cdots \to t_n$ is a quasi-cyclic top chain, starting from t_i with $t_i\theta = w_i \notin \mathcal{L}$.

Now theorem 5 directly results from property A.4 above and the definition of self-stabilization.