Distributed Computing manuscript No.
(will be inserted by the editor)

Proving Convergence of Self-Stabilizing Systems Using First-Order Rewriting and Regular Languages ⋆

J. Beauquier[1], B. Bérard[2], L. Fribourg[2] and F. Magniette[1]

[1] LRI, CNRS URA 410, Université Paris-Sud 91405 Orsay cedex, France
{jb,magniett}@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

The date of receipt and acceptance will be inserted by the editor

Summary. 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. In order to make the method terminate more often, we also propose an adapted procedure that manipulates "schemes", i.e. regular sets of words, and incorporate a process of scheme generalization. The interest of the method is illustrated on several nontrivial examples.

keywords: self-stabilization, rewriting systems.

1 Introduction

Introduced by Dijkstra, with three mutual exclusion algorithms on a ring of processes [11], the notion of self-stabilization has been largely studied for the last ten years (see [29,31] 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 component local states and it is characterized by a set $\mathcal{S}$ 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 at each step by the central demon, it is guaranteed to reach 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 proved 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 [31]. 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.[23,14,5]).

We propose here a new approach for proving the absence of cycle. Configurations are viewed as words of a formal language, transitions of $\mathcal{S}$ as rewrite rules, and the no-cycle property as a variant of the nontermination property for rewrite rules. The absence of infinite sequences will be shown by refining the generation procedure of *reduction chains*[1], first proposed by Dershowitz for proving string rewriting termination [8]. The method proposed here is new in that: 1) it uses a general technique of string-rewriting to deal with self-stabilization; 2) it does not consider all the possible rewrite derivations, but only "representative" ones, using a restricted first-order rewriting strategy. A generalization strategy, replacing sequences of words with regular languages, is also incorporated in the method in order to improve its termination.

Related work on self-stabilization proofs. [11] is without proof. In [12], a correctness proof is given for the third (3-state) algorithm of [11], by showing properties of executions using behavioral reasoning. As already pointed out, almost all further proof methods (cf. [31]) are based on norm functions (see an example in [19])

⋆ This paper is a revised and extended version of a communication given by the three first authors, at Symp. DISC'99, under the title "A new rewrite method for proving convergence of self-stabilizing systems" (LNCS 1693, Springer-Verlag, pp. 240–253).

[1] also called *forward closures* in [9].

but, as expressed by Gouda in [15]: "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 [3,16], attractor or staircase methods [15,29], automatic transformations into stabilizing systems [2]. The idea of representing sets of configurations as regular languages is used in [18], but, only at a "ground" level (without use of 1st-order variables). Note that, recently, some works were done for proving convergence without appealing to a norm function: [32] 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 [25–27] 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 explains how self-stabilization can be viewed as a property of string rewriting systems. In section 3, we give a basic procedure, inspired from Dershowitz, that, when it terminates, allows to decide self-stabilization. In order to make the procedure terminate more often, we incorporate a process of generalization into it (section 4). Section 5 shows application of the method on detailed examples and section 6 concludes with final remarks and perspectives.

2 Self-stabilizing systems as string rewrite systems

We first recall some basic definitions from (string) rewrite systems [10,6]. 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 a *(1st-order) 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 by $Dom(\theta)$. This mapping extends trivially to words and the result $\theta(w)$ is called an *instance* of the word w. 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 $W \in Dom(\theta)$.

2.1 String rewrite systems.

The string rewrite systems $\mathcal{S}$ considered here contain length-preserving rules, divided into three subsets: *top rules* in $Top_{\mathcal{S}}$ are applied to the rightmost part of words; *bottom rules* in $Bottom_{\mathcal{S}}$ are applied to the leftmost part of words (or simultaneously at both ends); the rest of rules in $Middle_{\mathcal{S}}$ 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,

$Middle_{\mathcal{S}}$ is made of rules of the form:
$\#X\ell Y\# \to \#XrY\#$
$Top_{\mathcal{S}}$ is made of rules of the form:
$\#X\ell\# \to \#Xr\#$
$Bottom_{\mathcal{S}}$ is made of rules of the form:
$\#\ell X\# \to \#rX\#$ or $\#\ell_1 X\ell_2\# \to \#r_1 Xr_2\#$.

We are going to apply these rules either to ground words or to 1st-order words of the form $\#uWv\#$ where u, v are strings over Σ^*, and W is a variable.

Example 1. Consider the following rules from Beauquier-Debas algorithm (see section 5):
B_1 : $\#2X1\# \to \#1X2\#$ is a bottom-rule.
T_4 : $\#X12\# \to \#X21\#$ is a top-rule.
M_1 : $\#X10Y\# \to \#X01Y\#$ and
M_4 : $\#X02Y\# \to \#X20Y\#$ are middle rules.

These rules are used throughout the next sections to illustrate the notions which are defined, as well as the method we propose.

Note that, strictly speaking, rules M_1 and M_4 can be applied only if $Y \neq \varepsilon$. For example, M_1 should correspond to 3 rules of the form $\#X10aY\# \to \#X01aY\#$ with $a \in \{0, 1, 2\}$. For the sake of conciseness, we neglect such a restrictive condition of application (except at section 5).

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

Example 2. Consider B_1 : $\#2X1\# \to \#1X2\#$. The ground word $w = \#2011\#$ is an instance of the lefthand side, using the substitution $\{X/01\}$.
Replacement with the righthand side yields the reduced form $w' = \#1012\#$.

Reduction of a ground word w using a middle rule R : $X\ell Y \to XrY$ merely consists in searching for a substring ℓ of R, and replacing it with r. Note that reduction can occur in various places (corresponding to all the possible positions where ℓ occurs as a substring of w).

2.2 Self-stabilization.

We are now able to give a formal definition of self-stabilization for a system modeled as a string rewrite system $\mathcal{S}$. 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 legitimate configurations*, as $\mathcal{L} = \bigcup_{N \geq 2} L_N$.

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

(0) *Each ground word is reducible via $\mathcal{S}$.*
(1) *$\mathcal{L}$ is* closed via *$\mathcal{S}$, i.e: $w \in \mathcal{L} \wedge w \rightarrow_{\mathcal{S}} w' \Rightarrow w' \in \mathcal{L}$, for all ground words w, w'.*
(2) *There is no ground cyclic derivation of the form $w_1 \rightarrow_{\mathcal{S}} \cdots \rightarrow_{\mathcal{S}} w_n = w_1$ with $w_1 \notin \mathcal{L}$.*

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 [7]).
Note that assuming (1), an equivalent version for (2) is:
(2′) There is no infinite ground derivation Δ via $\mathcal{S}$, such that $\Delta \cap \mathcal{L} = \emptyset$.

3 A first-order characterization of cycles

Assuming now that $\mathcal{S}$ 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 $\mathcal{S}$ (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 reduction chains, which is transposed from [8] in our particular context.

3.1 Minimal reductions and chains.

We now deal with one-variable words t of the form $\#uWv\#$, with $u, v \in \Sigma^*$ and $W \in \mathcal{V}$. The notion of ground reduction defined in section 2.1, extends trivially to such one-variable words: it suffices to consider W as a new constant (i.e., to extend Σ with $\{W\}$), and reduce the word using a rule $R : \lambda \rightarrow \rho$, as in section 2.1. We say in this case that reduction is done using substitution $\sigma = id$. Given a first-order word t and a rewrite rule R, it is also possible to consider nontrivial instantiations of W ($\sigma \neq id$), that make t reducible. This problem is a particular case of the unification problem: finding common instances of t and λ. The general unification problem for words is complex, and was solved by Makanin [28]. However our particular unification problem here is simple, because t and λ do not share variables, and are "linear" (i.e. contain at most one occurrence of the same variable). In such a case there exists a *complete set of minimal unifiers* that is finite: roughly speaking, it suffices to consider all the manners in which t and λ overlap depending on the possible instantiations of their variables (see, e.g., [21]). Assume given a minimal complete set of unifiers $\mu_1, ..., \mu_k$ ($t\mu_i = \ell\mu_i$ for $i = 1, ..., k$), each instance $t\mu_i$ of t reduces to $r\mu_i$ ($t\mu_i \rightarrow r\mu_i$). However, we disregard unifiers μ_j which instantiate t at a "variable position" (replacing variable W of t with a subword of the form $\#uW'v\#$ with $u, v \neq \varepsilon$), which will turn out to be unnecessary in our context (see remark, section 3.3). Such an operation of minimal reduction (at nonvariable position) is an adaptation of the operation of "narrowing" [30,13,20] in our context (one-variable words rather than 1st-order terms).

Example 3. Consider the rule $\#X11Y\# \rightarrow \#X22Y\#$. 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
$\mu_4 : \{W/W_1 11W_2\} \cup \{X/1W_1, Y/W_2 1\}$.

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:
$\#11W'1\# \rightarrow \#22W'1\#$, $\#1W'11\# \rightarrow \#1W'22\#$ and $\#11\# \rightarrow \#22\#$.

Let us now define in a formal and constructive manner the operation of minimal reduction (at nonvariable position). We distinguish two basic cases, depending on wether the involved substitution is identity ($\sigma = id$) or not. Suppose that we are given a middle rule $R : \#X\ell Y\# \rightarrow \#XrY\#$ where ℓ is of the form $a_1 \cdots a_n$ (with $a_i \in \Sigma$). The substitutions $\sigma \neq id$ involved in minimal reductions via R form the set $D_R = A_R \cup B_R \cup C_R$, with:

- A_R is the set of substitutions $\alpha_i : \{W/W'a_1 \cdots a_i\}$ for $1 \leq i \leq n-1$,
- B_R is the set of substitutions $\beta_i : \{W/a_{i+1} \cdots a_n W'\}$ for $1 \leq i \leq n-1$,
- C_R is the set of (ground) substitutions $\gamma_{i,j} : \{W/a_{i+1} \cdots a_j\}$ for $1 \leq i \leq j \leq n-1$ (with the convention that $\gamma_{i,j}$ is $\{W/\varepsilon\}$ if $i = j$).

A similar set of substitutions D_R can be defined for a top or bottom rule R.

Definition 2. *A word t is* minimally reducible *to u via rule $R : \lambda \rightarrow \rho$ using substitution $\sigma \in D_R \cup \{id\}$ (written: $t\sigma \rightarrow_R u$, or more simply $t\sigma \rightarrow u$) iff:*

- *$\sigma = id$, t is an instance of λ, and u is the corresponding instance of ρ.*
- *For a middle rule $R : \#X\ell Y\# \rightarrow \#XrY\#$ with $\ell = a_1 \cdots a_n$, and a word $t = \#t_1 W t_2\#$:*
  - *t_2 begins with string $a_{i+1} \cdots a_n$ (for some $1 \leq i \leq n-1$), $\sigma = \alpha_i$, and u is obtained from t by replacing $Wa_{i+1} \cdots a_n$ with $W'r$.*

- t_1 *ends with string* $a_1 \cdots a_i$ *(for some* $1 \le i \le n-1$*),* $\sigma = \beta_i$*, and* u *is obtained from* t *by replacing* $a_1 \cdots a_i W$ *with* rW'*.*
- t_2 *begins with* $a_{j+1} \cdots a_n$ *and* t_1 *ends with* $a_1 \cdots a_i$ *(for some* $1 \le i \le j \le n-1$*),* $\sigma = \gamma_{i,j}$*, and* u *is obtained from* t *by replacing* $a_1 \cdots a_i W a_{j+1} \cdots a_n$ *with* r*.*

- *For a top or bottom rule* R*, minimal reduction of* t *(using* $\sigma \neq id$*), is defined similarly to the case of a middle rule.*

Example 4. For rule $M_1 : \#X10Y\# \to \#X01Y\#$ and word $t_1 = \#W02\#$, there is a single substitution $\alpha_1 : \{W/W'1\}$ that yields a minimal reduction.
This gives $\#W'102\# \to \#W'012\#$.

For rule $R : \#X100Y\# \to \#X110Y\#$ and word $t_2 = \#W003\#$, there are two possible instantiations $\alpha_1 : \{W/W'10\}$ and $\alpha_2 : \{W/W'1\}$, which yield respectively the following minimal reductions: $\#W'10003\# \to \#W'11003\#$ and $\#W'1003\# \to \#W'1103\#$.

We can now define the notion of "minimal reduction chains". They are direct transpositions of definitions or properties of [8] in our particular context.

Definition 3. *The* minimal top reduction chains *of a rewrite system* $\mathcal{S}$ *form a set of derivations inductively defined as follows:*

- *Every top rule of* $\mathcal{S}$ *is a minimal top reduction chain.*
- *If* $C : \ t_0 \to \cdots \to t_n$ *is a top minimal reduction chain and* R *is a rule of* $\mathcal{S}$ *such that* $t_n\sigma \to u$ *via* R*, then* $t_0\sigma \to \cdots t_n\sigma \to u$ *is a top minimal reduction chain, called a* successor *of* C *via* R *using* σ*.*

The transitive closure of the successor relation is called "iterated successor". Henceforth, we will simply say "top chain" instead of "top minimal reduction chain".

Example 5. Consider the rule $T_4 : \ \#W12\# \to \#W21\#$ viewed as a chain. Its successor via $B_1 : \ \#2X1\# \to \#1X2\#$ using $\{W/2W'\}$ is:
$C_1 : \ \#2W'12\# \to \#2W'21\# \to \#1W'22\#$.

Recall that in all rules $\lambda \to \rho$ of the string rewrite systems considered here, the substring ℓ of λ is replaced by substring r of ρ, which has the same length. Therefore, a chain $t_1 \to t_2 \to ... \to t_n \ldots$ is such that either all the words $t_1, ..., t_n$ are ground and of the same length, or each t_i is of the form $\#u_i W v_i\#$ where the u_is and v_is are strings of Σ^* of same length. For the same reason of length preservation, there must exist $1 \le i < j$ such that $t_i = t_j$, within any infinite top chain (ground or not). This explains the use of the following definition.

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*.*

Example 6. Consider the alphabet $\Sigma = \{0, 1\}$ and the following system $\mathcal{T}$:

Bottom	C_0 :	$\#0X0\# \to \#1X0\#$	
	C_1 :	$\#1X1\# \to \#0X1\#$	
Top	U_0 :	$\#X01\# \to \#X00\#$	
	U_1 :	$\#X10\# \to \#X11\#$	
Middle	N_0 :	$\#X01Y\# \to \#X00Y\#$	(with $Y \neq \varepsilon$)
	N_1 :	$\#X10Y\# \to \#X11Y\#$	(with $Y \neq \varepsilon$)

Let $\mathcal{L}$ be: $\#0^+1^+\# \cup \#1^+0^+\# \cup \#11^+\# \cup \#00^+\#$ and consider the top rule $U_0 : \ \#W01\# \to \#W00\#$ viewed as a chain. Its successor via N_1 using $\{W/W'1\}$ is: $\#W'101\# \to \#W'100\# \to \#W'110\#$.
The successor via N_0 using $\{W'/W''0\}$ is then:
$\#W''0101\# \to \#W''0100\#$
$\to \#W''0110\# \to \#W''0010\#$.
Now the successor via B_0 using $\{W''/\varepsilon\}$ is:
$t_1 = \#0101\# \to t_2 = \#0100\# \to t_3 = \#0110\# \to t_4 = \#0010\# \to t_5 = \#1010\#$.
Note that this last chain is ground. It is easy to see that it can be prolonged by the following sequence of reductions:
$t_5 = \#1010\# \to t_6 = \#1011\# \to t_7 = \#1001\# \to t_8 = \#1101\# \to t_9 = \#0101\#$.
A quasi-cyclic chain $t_1 \to \cdots \to t_9 = t_1$ has thus been obtained.

3.2 A characterization of self-stabilization.

We can now state our main result:

Theorem 1. *Let* $\mathcal{S} = Middle_{\mathcal{S}} \cup Top_{\mathcal{S}} \cup Bottom_{\mathcal{S}}$ *be a rewrite system and let* $\mathcal{L}$ *be a set of configurations. If*

(0) each ground word is reducible via $\mathcal{S}$*,*
(1) $\mathcal{L}$ *is closed via* $\mathcal{S}$*, and*
(3) $\mathcal{S} - Top_{\mathcal{S}}$ *is terminating.*

Then $\mathcal{S}$ *is self-stabilizing w.r.t.* $\mathcal{L}$ *iff there is no quasi-cyclic top chain* $t_1 \to \cdots \to t_n$ *via* $\mathcal{S}$*, such that* $u_n \notin \mathcal{L}$ *for some ground instance* u_n *of* t_n*.*

Example 7. Let us illustrate theorem 1 on the system $\mathcal{T}$ of the example above. First, it is easy to see that $\mathcal{T}$ and $\mathcal{L}$ satisfy properties $(0), (1)$ and (3). From the existence of quasi-cyclic top chain $t_1 \to \cdots \to t_9$ with $t_1 = t_9 = \#0101\# \notin \mathcal{L}$, it now follows by theorem 1 that $\mathcal{T}$ is *not* self-stabilizing w.r.t. $\mathcal{L}$.

Theorem 1 is an adaptation of Dershowitz's theorem ([8], p. 454) characterizing nonterminating string rewrite systems as those having at least one cycling chain (or infinitely many noncycling infinite chains, which cannot happen in our case).

Remark. In theorem 1, we require the termination property of $\mathcal{S} - Top_{\mathcal{S}}$ (condition (3)). This condition is new w.r.t. original Dershowitz's framework of [8]. Condition (3) allows us to refine Dershowitz's method and to focus on *top* chains (instead of chains starting with arbitrary rules), so that the number of chains generated is further restricted. In practice, condition (3) is often satisfied when dealing with stabilizing systems having infinite executions, that means solving dynamical problems (like token circulation, networks traversal, resource allocation), because these systems, from their very specification, have to be fair in the following sense (cf, e.g., [31]): any infinite execution involves any process (and in particular the top process) an infinite number of times. It is easy to see that a fair system always satisfies (3). Details about how to mechanically check (3), are given in [4].

3.3 Proof of theorem 1

The proof of theorem 1 relies on properties involving the notions of "active" or "inactive" steps within infinite ground derivations, as introduced by Dershowitz [8].

Definition 5. *The* active area *of a ground word w_i in a ground derivation $w_1 \to w_2 \to \cdots \to w_n$ is the 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.*

Definition 6. *An* active ground derivation *via $\mathcal{S}$ (resp.* inactive ground derivation *via $\mathcal{S}$) is a ground derivation $w_1 \to w_2 \to \cdots \to w_n$ in which rules of $\mathcal{S}$ are applied only in the active area (resp. inactive area) of words. We denote by $\xrightarrow{act}$ (resp. $\xrightarrow{inact}$) an application of a rule at an active area (resp. inactive area) of a ground word.*

In the derivations, we start with a single active top letter and the successive reductions will increase the active part in the words. Active letters will be put in bold.

Example 8. Starting from the ground word $\#201\mathbf{2}\#$ and applying successively top rule $T_4 : \#X12\# \to \#X21\#$ and bottom rule $B_1 : \#2X1\# \to \#1X2\#$ at active area, we obtain the following active ground derivation $\Delta : \#201\mathbf{2}\# \xrightarrow{act} \#20\mathbf{21}\# \xrightarrow{act} \#\mathbf{1}0\mathbf{22}\#$.

The following property is the counterpart of the relation between reduction sequences and narrowing sequences in first-order term theory [20]. The proof is analogous, therefore omitted.

Lemma 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).*

Remark. The lemma states that any ground derivation is an instance of some 1st-order top chain. Such a chain is obtained by using a sequence of minimal instantiations $\{\sigma_1, \cdots, \sigma_n\}$, where σ_i is of the form $\{W/uW'\}$, $\{W/W'u\}$ or $\{W/u\}$.
This 1st-order covering holds in spite of the fact that instantiations σ of the form $\{W/uW'v\}$ with $u, v \neq \varepsilon$ are discarded by definition. This justifies *a posteriori* our focus on minimal reductions at *nonvariable positions*.

Example 9. The active ground derivation (see above)
$\Delta : \#201\mathbf{2}\# \xrightarrow{act}_{T_4} \#20\mathbf{21}\# \xrightarrow{act}_{B_1} \#\mathbf{1}0\mathbf{22}\#$
is an instance (via the ground substitution $\{W'/0\}$) of the top chain
$C_1 : \#2W'12\# \to \#2W'21\# \to \#1W'22\#$.

Finally we will use the following property (also used by Dershowitz in a more general context [8]):

Lemma 2 (semi-commutation lemma). *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 $w_1 \xrightarrow{inact} w_2' \xrightarrow{act} w_3$.*

Example 10. Starting from the word $\#101\mathbf{2}\#$, consider the following derivation via T_4 then M_1:
$\#101\mathbf{2}\# \xrightarrow{act}_{T_4} \#10\mathbf{21}\# \xrightarrow{inact}_{M_1} \#01\mathbf{21}\#$.
The order of rule application can be permuted to get:
$\#101\mathbf{2}\# \xrightarrow{inact}_{M_1} \#011\mathbf{2}\# \xrightarrow{act}_{T_4} \#01\mathbf{21}\#$.

Therefore inactive steps can always be switched with active steps, and pushed upwards. It remains to show that the number of inactive steps in infinite ground derivations is necessarily finite.

Proposition 1. *Let $\mathcal{S}$ be a rewrite sytem such that $\mathcal{S} - Top_{\mathcal{S}}$ is terminating. Then any infinite ground derivation via $\mathcal{S}$ 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 $\mathcal{S}$. 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 $x_nv_ny_n$, 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 do not involve 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 $\mathcal{S} - Top_{\mathcal{S}}$ is terminating. The only possible case is therefore (a): all the inactive subareas disappear after a finite number of steps.

Finally, all inactive steps from an infinite ground derivation can be pushed upwards, thus yielding a purely active suffix, which is an instance of a top chain. Formally:

Proposition 2. *Suppose that $\mathcal{S} - Top_{\mathcal{S}}$ is terminating and $\mathcal{L}$ is closed via $\mathcal{S}$. Then there is an infinite ground derivation via $\mathcal{S}$ containing no word in $\mathcal{L}$ if and only if there is a quasi-cyclic top chain via $\mathcal{S}$, 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 not in $\mathcal{L}$. By property 1 (since $\mathcal{S} - Top_{\mathcal{S}}$ is terminating), this infinite derivation contains only a finite number of inactive steps. Applying iteratively the semi-commutation lemma (2), one can push back these inactive steps to the beginning of the derivation, thus obtaining a reordered infinite ground derivation Δ'. From some point on, there are only active steps in derivation Δ'. Let $\Delta'' : w_i \to w_{i+1} \to \cdots$ denote this active infinite ground part of derivation not in $\mathcal{L}$. 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 (1), there is a 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 chain, ending at t_n with $t_n\theta = w_n \notin \mathcal{L}$.

Now, theorem 1 directly results from proposition 2 and the definition of self-stabilization.

3.4 Basic procedure and φ-refinement

Theorem 1 suggests to prove self-stabilization by the following basic procedure:

- Generate all top chains $t_1 \to \cdots \to t_n$ via $\mathcal{S}$, until $t_n = t_i$ for some $i < n$.
- If the only infinite chains generated are of the form $t_1 \to \cdots \to t_i \to \cdots t_n = t_i$ with all instances of t_n in $\mathcal{L}$, then $\mathcal{S}$ is self-stabilizing. Otherwise, $\mathcal{S}$ is not self-stabilizing.

In order to prove self-stabilization, it is then (necessary and) sufficient to generate only 1st-order sequences $t_1 \to \cdots \to t_i \to \cdots t_n$ that extend top rules, instead of blindly generating all the possible ground derivations, starting from arbitrary ground words. The generation procedure is restrictive because the procedure is first-order and manipulates one-variable words of the form $\#uWv\#$, instead of all the (infinite) sets of their ground instances. It is also restricted by the fact that sequences always start with the left-hand side of a top rule.

Theorem 1, and the associated procedure, can be refined by exploiting a measure, say φ, over words, which "never increases" when applying a rule i.e. such that: $w \to w' \Rightarrow \varphi(w) \geq \varphi(w')$. This is a relaxed assumption, with respect to norms that, as required in traditional self-stabilization proof methods, must "always decrease" (i.e., roughly speaking, norms ψ such that $w \to w' \Rightarrow \psi(w) > \psi(w')$). In addition with such a non-increasing mesure φ, we assume given a *φ-preserving version* $\mathcal{S}'$ of $\mathcal{S}$, that is a system $\mathcal{S}' = Middle_{\mathcal{S}'} \cup Bottom_{\mathcal{S}'} \cup Top_{\mathcal{S}'}$ such that: $w \to_{\mathcal{S}'} w'$ iff $w \to_{\mathcal{S}} w' \wedge \varphi(w) = \varphi(w')$. Now in any infinite derivation via $\mathcal{S}$, all the rewrite steps, after a finite number of them, are necessarily φ-preserving (because of the non-increasing property), and may be viewed as rewrite steps via $\mathcal{S}'$. In order to prove self-stabilization of $\mathcal{S}$, it becomes necessary and sufficient to show the absence of quasi-cyclic top chains, except those ending with an instance of $\mathcal{L}$, via $\mathcal{S}'$ (instead of $\mathcal{S}$) as far as $\mathcal{S}' - Top_{\mathcal{S}'}$ (instead of $\mathcal{S} - Top_{\mathcal{S}}$) is terminating. More precisely, theorem 1 becomes:

Theorem 2. *Let $\mathcal{S} = Middle_{\mathcal{S}} \cup Top_{\mathcal{S}} \cup Bottom_{\mathcal{S}}$ be a rewrite system and $\mathcal{L}$ a set of configurations. Let φ be a non-increasing mesure and $\mathcal{S}' = Middle_{\mathcal{S}'} \cup Top_{\mathcal{S}'} \cup Bottom_{\mathcal{S}'}$ a φ-preserving version of $\mathcal{S}$. If*

(0) each ground word is reducible via $\mathcal{S}$,
(1) $\mathcal{L}$ is closed via $\mathcal{S}$, and
(3') $\mathcal{S}' - Top_{\mathcal{S}'}$ is terminating.

Then $\mathcal{S}$ is self-stabilizing w.r.t. $\mathcal{L}$ iff there is no quasi-cyclic top chain $t_1 \to \cdots \to t_n$ via $\mathcal{S}'$, such that $u_n \notin \mathcal{L}$ for some ground instance u_n of t_n.

Example 11. Consider the middle rules of Ghosh's algorithm (see Section 5 for details):

$$M_1 : \ \#X(q+1)qY\# \to \#X(q+1)(q+1)Y\#$$
$$M_2 : \ \#Xq(q+1)Y\# \to \#X(q+1)(q+1)Y\#$$

where $q \in \{0, 1, 2, 3\}$ and '+' is addition modulo 4.

The convergence proof uses a norm function (Br, Ds) such that either Br or Ds *strictly* decreases at each step, but individually Br and Ds are only non-increasing functions. While Ds is a very subtle function, Br is simply the number of *breaks*, i.e. the number of neighbouring states q, q' of the string which differ by at least one unit. In contrast, our method proves the convergence, with the help of measure Br only: we will focus on Br-preserving infinite derivations, i.e. infinite derivations which preserve the number of breaks. Since Br is non-increasing and bounded, any infinite derivation has an infinite Br-preserving suffix, so there is no loss of generality.
To obtain the system $\mathcal{S}'$, we modify the rules above into Br-preserving rules as follows:

$$M_1 : \ \#X(q+1)qqY\# \to \#X(q+1)(q+1)qY\#$$
$$M_2 : \ \#Xqq(q+1)Y\# \to \#Xq(q+1)(q+1)Y\#$$

It is easy to show that $\mathcal{S}'$ is a Br-preserving version of $\mathcal{S}$ ($w \to_{\mathcal{S}'} w'$ iff $w \to_{\mathcal{S}} w' \wedge Br(w) = Br(w')$) and that $\mathcal{S}' - Top_{\mathcal{S}'}$ is terminating.

Remark. It is sometimes convenient to use a measure φ that is "never decreasing", instead of never increasing. The enhanced theorem still holds, because again, in any infinite derivation via $\mathcal{S}$, all the steps, after a finite number of them, must be φ-preserving (provided that φ is bounded upward, e.g., by the number N of machines the ring is made of). Such a reasoning is used by Burns and Pachl (see [7], p. 339), who focus on infinite derivations preserving the number of "dynamic segments" of configurations. We also use a non-decreasing mesure φ in the example of section 5.3.

Henceforth, when given a measure φ and an associated system $\mathcal{S}'$, we implicitly focus on the generation of top chains via $\mathcal{S}'$ instead of $\mathcal{S}$. This allows us to restrict even more the number of generated chains. However, in practice, in spite of this refinement, the generation procedure does not terminate, but yields infinitely many chains. To solve this problem, we introduce in section 4 a notion of chains over *regular sets* of words (instead of simply words) combined with a notion of "generalization".

4 A sufficient condition for self-stabilization

As mentioned earlier, top chains, when generated in a brute manner, are frequently in infinite number. However it is often possible to discover some recurrent forms for words t_n appearing at the end of chains. Consider again the subset of rules $\mathcal{S}' = \{T_4, B_1, M_1, M_4\}$ from Beauquier-Debas system:

$T_4 : \#X12\# \to \#X21\#$,
$B_1 : \#2X1\# \to \#1X2\#$,
$M_1 : \#X01Y\# \to \#X10Y\#$,
$M_4 : \#X02Y\# \to \#X20Y\#$.

Starting from T_4, and applying iteratively rule M_4, one generates chains of the form:

$\#W12\# \to \#W21\#$,
$\#W012\# \to \#W021\# \to \#W201\#$,
$\#W0012\# \to \#W0021\# \to \#W0201\# \to \#W2001\#$,
$\cdots$

These chains are in infinite number, but all of them (except the first one) end with words of the form $\#W20^j1\#$ with $j > 0$. It is then convenient to generalize words of the form $\#W20^j1\#$ by the regular set $\#W20^+1\#$, and replace first-order derivations over words by first-order derivations over regular sets. Such regular sets are called "schemes" in the following. Derivations over schemes are then defined in such a manner that they "cover" all the possible top chains over words (overapproximation). These generalized derivations may be represented under the compact form of paths along the nodes of a symbolic graph, each node corresponding to a scheme. Absence of infinite derivations (i.e., self-stabilization) is then shown by checking that every path of the graph uses only a bounded number of top rules. (This exploits the assumption of termination for $\mathcal{S} - Top_{\mathcal{S}}$, see section 4.3.) We claim that such a process of generalization improves the convergence of the method of top chain generation.

4.1 Replacing words by schemes

We are now going to define the notion of "schemes", an appropriate form of regular languages that contain sets of words considered before. We extend accordingly our notion of reduction chains by manipulating schemes instead of words.

Definition 7. *A* first-order scheme *S is a language of the form $\#LWM\#$ where L, M are (nonempty) regular languages over Σ^*, and W a first-order variable. A* ground scheme *S is a language of the form $\#L\#$ where L is a regular language over Σ^*. A* scheme *is either a 1st-order or a ground scheme.*

In the following we assume that the set of legitimate configurations $\mathcal{L}$ is expressed as a ground scheme.

Note that any first-order word of the form $\#uWv\#$ (with strings u, v) can be seen as a special first-order scheme with $L = \{u\}$ and $M = \{v\}$. Likewise any ground word is a special ground scheme.

4.2 Minimal reductions of schemes and generalization

The notion of minimal reduction over words extends naturally over schemes. Formally:

Definition 8. *Let S be a scheme, R a rule and σ a substitution of $D_R \cup \{id\}$. The* reduced form of S via R using *σ is the scheme S' defined by:*
$S' = \{s' \mid \exists s \in S \text{ such that } s\sigma \to_R s'\}$.
This is written $S\sigma \to_R S'$.

Example 12. For rule $M_4 : \#X02Y\# \to \#X20Y\#$, the scheme $S = \#W20^+1\#$ minimally reduces via M_4 to $S' = \#W'200^+1\#$ using $\sigma : \{W/W'0\}$. The reduction is written $S\sigma = \#W'020^+1\# \to \#W'200^+1\#$.

Generalization over 1st-order schemes is an overapproximation of the regular languages surrounding the 1st-order variable of a scheme. More precisely, a 1st-order scheme $T : \ \#M_1WM_2\#$ is a *generalization* of a 1st-order scheme $S : \ \#L_1W'L_2\#$ iff $L_1 \subseteq M_1$ and $L_2 \subseteq M_2$. Note that the names of variables appearing in T and S (viz., W and W') do not matter. Henceforth, without loss of generality, we always rename first-order variables $W', W'', \cdots$ appearing in schemes, to W. In particular a substitution of the form $\{W/W'u\}$ is simply written $\{W/Wu\}$. With this convention, a generalization T of 1st-order scheme S is just a 1st-order scheme that contains S, i.e., such that: $S \subseteq T$. Likewise, we say that a ground scheme T is a generalization of a ground scheme S iff $S \subseteq T$.

In the following, we interleave the generalization process with minimal reduction. Given a rule R, and a substitution $\sigma \in D_R \cup \{id\}$, we say that a scheme T is a *generalized successor* of scheme S, and write $S\sigma \nearrow_R T$ (or more simply $S\sigma \nearrow T$), iff $S\sigma \to_R S'$ and $S' \subseteq T$.

Example 13. For rule $M_4 : \#X20Y\# \to \#X02Y\#$ and schemes $S = \#W20^+1\#$ and $S' = \#W200^+1\#$, we have $S\sigma \to S' \subseteq S$ (with $\sigma : \{W/W0\}$). We can thus write $S\sigma \nearrow S$. So S is its own generalized successor via M_4 using σ.

We now express formally the correspondence between relation $\nearrow$ at the scheme level and $\to$ at the word level. We have:

Lemma 3. *Given a rule R and a substitution $\sigma \in D_R \cup \{id\}$, we have, for any scheme S and any word $s \in S$:*
$s\sigma \to_R t$ implies $S\sigma \nearrow_R T$ for some scheme T such that $t \in T$.

We now give formally the notion of "generalized top chain".

Definition 9. *The* generalized minimal top chains *of a rewrite system $\mathcal{S}$ form a set inductively defined as follows:*

- *Every top rule $t_0 \to t_1$ of $\mathcal{S}$ is a generalized chain, written $\{t_0\} \nearrow \{t_1\}$.*
- *If G : $T_0 \nearrow \cdots \nearrow T_n$ is a generalized minimal top chain and R is a rule of $\mathcal{S}$ such that $T_n\sigma \nearrow T_{n+1}$, then $T_0\sigma \nearrow \cdots \nearrow T_n\sigma \nearrow T_{n+1}$ is a generalized minimal top chain called* generalized successor *of G (via R using σ).*

Lemma 3 generalizes as follows:

Lemma 4. *Given a top chain over words,*
$t_0\sigma_1 \ldots \sigma_n \to_{R_1} \cdots \to_{R_{n-1}} t_{n-1}\sigma_n \to_{R_n} t_n$,
via rules $R_1, \ldots, R_n$ of $\mathcal{S}$, using substitutions $\sigma_1, \ldots, \sigma_n$, and a scheme T_0 such that $t_0 \in T_0$, there is a top chain over schemes,
$T_0\sigma_1 \ldots \sigma_n \nearrow_{R_1} \cdots \nearrow_{R_{n-1}} T_{n-1}\sigma_n \nearrow_{R_n} T_n$,
using the same substitutions, such that $t_i \in T_i$ for $i = 1, \ldots, n$.

This lemma states that any top chain at the word level is "covered" by a corresponding top chain at the scheme level. Hereafter, we describe a procedure of chain generation at the scheme level. This procedure has a top rule $t_0 \to t_1$ as input and Γ denotes the set of schemes for which the successors remain to be computed.

Generalized chain generation($t_0 \to t_1$)
Initially: $\Gamma = \{t_1\}$.
While $\Gamma \neq \emptyset$
do
1. Select $S \in \Gamma$.
2. Compute the finite set of generalized successors, say $\{S^i_{new}\}_i$, of S
($S\sigma \nearrow_R S^i_{new}$ for some rule $R \in \mathcal{S}$ and some $\sigma \in D_R \cup \{id\}$).
3. For all i, add S^i_{new} to Γ unless $S^i_{new} \subseteq S_{old}$ for some S_{old} in Γ.
4. $\Gamma := \Gamma - \{S\}$.
od

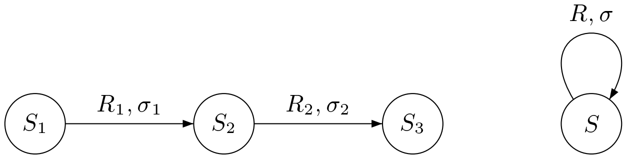

Fig. 1. Graphs for reductions $S_1\sigma_1\sigma_2 \nearrow_{R_1} S_2\sigma_2 \nearrow_{R_2} S_3$ and $S\sigma \nearrow_R S$

Note that test $S^i_{new} \subseteq S_{old}$ is decidable since it deals with inclusion of regular languages. An optimization of the above procedure (that will be implicit henceforth) consists in computing successors of S only if S is *not* a subset of $\mathcal{L}$. This is justified, since we are only interested in detecting the existence of infinite chains that do *not* intersect with $\mathcal{L}$ (*cf.* criterion ($2'$), Section 2.2).

Example 14. As already seen above for Beauquier-Debas system, the righthand side $\#W21\#$ of top rule T_4 minimally reduces via M_4 to scheme $S_1 = \#W20^+1\#$, which is its own generalized successor via M_4 using $\{W/W0\}$. On the other hand, scheme $S_1 = \#W20^+1\#$ is minimally reducible via B_1 (using $\{W/2W\}$) to
$U = \#1W20^+2\#$.
Scheme U is itself minimally reducible via M_1 (using $\{W/0W\}$) in an iterative way, which yields
$\#01W20^+2\#$, $\#001W20^+2\#$, $\#0001W20^+2\#$, etc.
Scheme U can be thus generalized as
$S_2 = \#0^*1W20^+2\#$,
which is thus a generalized successor of S_1 via B_1 (using $\{W/2W\}$).
The latter scheme is minimally reducible via M_4 (resp. M_1), but this yields only the subset $\#0^+1W200^+2\#$ (resp. $\#0^+01W20^+2\#$) of S_2. Therefore S_2 is its own generalized successor via M_4 (using $\{W/W0\}$) and via M_1 (using $\{W/0W\}$).

4.3 Graph construction

It is convenient to represent a minimal reduction of the form $S_1\sigma_1 \nearrow_{R_1} S_2$ under the form of an edge, labelled (R_1, σ_1), from S_1 to S_2. Likewise, a chain of the form $S_1\sigma_1\sigma_2 \nearrow_{R_1} S_2\sigma_2 \nearrow_{R_2} S_3$ is represented as an edge (labelled (R_1, σ_1)) from S_1 to S_2, followed by an edge (labelled (R_2, σ_2)) from S_2 to S_3 (see the graph on the left in figure 1). If additionally, schemes are represented as labels of nodes and structure sharing is used in order to merge the representation of nodes associated with identical schemes, then the generation of scheme successors corresponds to the construction of a graph, where paths between nodes represent chains over schemes. For example, the graph on the right in figure 1 corresponds to the minimal reduction $S\sigma \nearrow_R S$ and represents under a compact form an infinite number of chains $S\sigma \nearrow_R S$, $S\sigma^2 \nearrow_R S\sigma \nearrow_R S, \ \ldots$

The process of graph construction, described below, takes a top rule $t_0 \to t_1$ as an input and builds iteratively generalized successors under the form of growing paths. Each node N of the graph is labelled with a scheme S, and referred to as pair (N,S). Each generalized top chain using substitutions $\sigma_1,\ldots,\sigma_n$, of the form $(U_0 \nearrow_{R_1}) \; U_1 \nearrow_{R_2} \cdots \nearrow_{R_n} U_n$, is represented as a path of the graph of the form $\langle (N_1,S_1),\ldots,(N_n,S_n)\rangle$, where (R_i,σ_i) labels the edge from N_i to N_{i+1} $(1 \leq i \leq n-1)$, such that $S_i\sigma_i\cdots\sigma_n = U_i$ $(1 \leq i \leq n-1)$ and $S_n = U_n$. Formally, the procedure is as follows:

Graph construction $(t_0 \to t_1)$
Initially: $Q = \{(N_1,\{t_1\})\}$.
While $Q \neq \emptyset$
do
1. Select $(N,S) \in Q$
2. Compute the finite set of successors of S, say $\{S^i_{new}\}_i$, ($S\sigma \nearrow_R S^i_{new}$ for some rule $R \in \mathcal{S}$ and some $\sigma \in D_R \cup \{id\}$).
3. For all i:
  3a. If $S^i_{new} \subseteq S_{old}$ for some node $(N_{old},S_{old}) \in Q$, add an edge, labelled with (R,σ), from (N,S) to (N_{old},S_{old}).
  3b. Otherwise, add node (N^i_{new},S^i_{new}) to Q, and edge, labelled (R,σ), from (N,S) to (N^i_{new},S^i_{new}).
4. $Q := Q - \{(N,S)\}$.
od

According to the optimization mentioned previously, successors of S, at step 2, are implicitly computed only if S is *not* a subset of $\mathcal{L}$. Therefore in the graph, no edge exits from nodes labelled with (subsets of) $\mathcal{L}$. Another obvious optimization consists in skipping step $3a$ in case an edge, labelled σ, already exists from (N,S) to (N_{old},S_{old}).

Example 15. The graph for generalized chain generation of Beauquier-Debas system $\mathcal{S}'$, with T_4 as an input, is depicted in figure 2.

From the construction of the graph, we have:

Proposition 3. *Suppose that during the scheme chain generation, there is a generated chain from $U_1 = \{t_1\}$ to U_n via $R_1,\ldots,R_{n-1}$, using $\sigma_1,\ldots,\sigma_{n-1}$:*
  $t_1\sigma_1\ldots\sigma_{n-1} \nearrow_{R_1} \cdots \nearrow_{R_{n-1}} U_n$).
Then, there is a path from $(N_1,\{t_1\})$ to a node of the form (N,U_n) via edges labeled $(R_1,\sigma_1),\ldots,(R_{n-1},\sigma_{n-1})$.

This gives a sufficient condition for self-stabilization:

Theorem 3. *If, for each top rule $t_0 \to t_1 \in \mathcal{S}$ as an input, there is no path in the associated graph, that uses $Top_{\mathcal{S}}$ infinitely often (apart from paths passing by $\mathcal{L}$), then $\mathcal{S}$ is self-stabilizing.*

Proof. Suppose there is no path using $Top_{\mathcal{S}}$ infinitely often (except paths passing by $\mathcal{L}$). Then, by proposition 3, there is no chain over schemes using $Top_{\mathcal{S}}$ infinitely often (except chains reaching $\mathcal{L}$). Now, by lemma 4, there is no chain over words using $Top_{\mathcal{S}}$ infinitely often (except chains reaching $\mathcal{L}$). Finally, using the fact that $\mathcal{S} - Top_{\mathcal{S}}$ terminates, there is no infinite chain over words (except those reaching $\mathcal{L}$), hence no quasi-cyclic chain $t_0 \to t_1 \to \cdots \to t_i \to \cdots \to t_n = t_i$ (except if $t_n \in \mathcal{L}$). Therefore, $\mathcal{S}$ is self-stabilizing.

In contrast with theorem 1, the condition above is no longer necessary, and we cannot deduce *non* self-stabilization in case an infinite chain is produced. On the other hand, we claim that the procedure terminates more often than its counterpart over words, thus allowing to prove self-stabilization in more cases.

5 Examples

In order to illustrate the method of section 4 we explain how it applies on three examples. (The last two of them can be skipped by the reader without loss of continuity.)

5.1 Beauquier-Debas algorithm

This system originates from [5], and is an adaptation of Dijkstra's third (3-state) algorithm [11]. In our formalism, it corresponds to the following system $\mathcal{S}$:

Bottom	B_1 :	$\#2X1\# \to \#1X2\#$	
Top	T_1 :	$\#X00\# \to \#X21\#$	
	T_2 :	$\#X10\# \to \#X01\#$	
	T_3 :	$\#X20\# \to \#X11\#$	
	T_4 :	$\#X12\# \to \#X21\#$	
	T_5 :	$\#X22\# \to \#X01\#$	
Middle	M_1 :	$\#X10Y\# \to \#X01Y\#$	(with $Y \neq \varepsilon$)
	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 $\mathcal{S}$. It is easy to check that any ground word (with a null sum of elements) is reducible via $\mathcal{S}$, and that $\mathcal{L}$ is closed via $\mathcal{S}$ (see [5]). Therefore, $\mathcal{S}$ is self-stabilizing iff there is no ground cyclic derivation via $\mathcal{S}$ containing an element $w \notin \mathcal{L}$. As remarked in [5], one can see that T_1,T_2,T_3 are applied at most once. As a consequence $\mathcal{S}$ is self-stabilizing iff there is no ground cyclic derivation via $\mathcal{S}_0 \equiv \mathcal{S} - \{T_1,T_2,T_3\}$ 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 $\mathcal{S}_0$. Besides, among rules of $\mathcal{S}_0$ only rules B_1,M_1,M_4,T_4 preserve the number of nonnull elements. The φ-refinement of the basic procedure thus consists in generating top chains via $\mathcal{S}' \equiv \{B_1,M_1,M_4,T_4\}$ instead of $\mathcal{S}_0$.

The graph constructed in figure 2 gives a complete picture of the situation illustrated in the previous examples. Since there is no infinite path using Top (T_4 is used

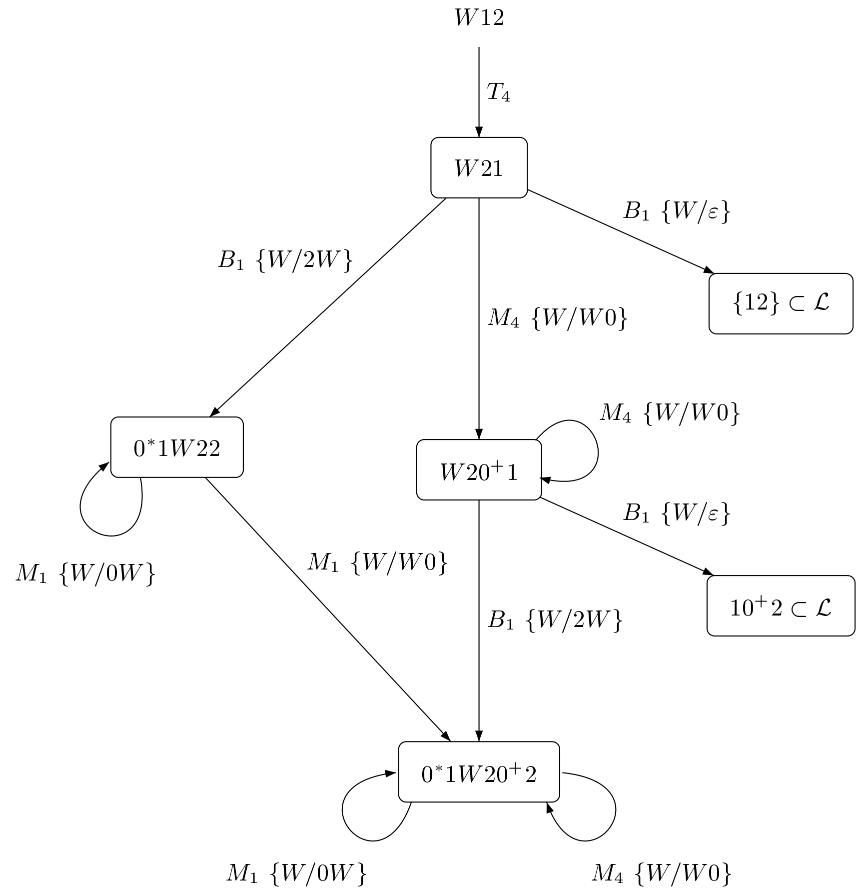

Fig. 2. The graph construction for T_4

at most once), it follows that there is no quasi-cyclic top chain $t_1 \to \cdots \to t_n$ via $\mathcal{S}'$, such that $u_n \notin \mathcal{L}$ for some ground instance u_n of t_n. Self-stabilization is thus proved for Beauquier-Debas's variant of Dijkstra's 3-state algorithm.

5.2 Ghosh's 4-state algorithm

Self-stabilization of Ghosh's algorithm [14], a variant of Dijkstra's 4-state algorithm, can be proved formally along the same lines. 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. $\{1, 3\}$). As explained in Section 2, 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 $\mathcal{S} = Middle \cup Top \cup Bottom$ of rewrite rules.

Middle

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

$M_2: \quad \#Xq(q+1)Y\# \to \#X(q+1)(q+1)Y\#$

where $q \in \{0, 1, 2, 3\}$ and '+' is addition modulo 4.

Top

$T_1: \quad \#X32\# \to \#X30\#$

$T_2: \quad \#X10\# \to \#X12\#$

Bottom

$B_1: \quad \#12X\# \to \#32X\#$

$B_2: \quad \#30X\# \to \#10X\#$

$\mathcal{L}$ is defined as $\#\{1^+, 3^+\}\{0^+, 2^+\}\#$.

As mentioned in section 3.4, Ghosh proves the convergence by considering a norm function (Br, Ds) such that either Br or Ds *strictly* decreases at each step. Recall that 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 and Ds measures 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, while our method uses measure Br only and focus on Br-preserving infinite derivations.

The Br-preserving version $\mathcal{S}'$ of $\mathcal{S}$ (cf. section 3.4) is obtained by modifying the middle rules as follows:

$M_1: \ \#X(q+1)qqY\# \to \#X(q+1)(q+1)qY\#$

$M_2: \ \#Xqq(q+1)Y\# \to \#Xq(q+1)(q+1)Y\#$

The graph construction is illustrated in figure 3 in the case of initial rule $T_1: \ \#W32\# \to \#W30\#$. Similarly, another representative graph can be obtained, starting

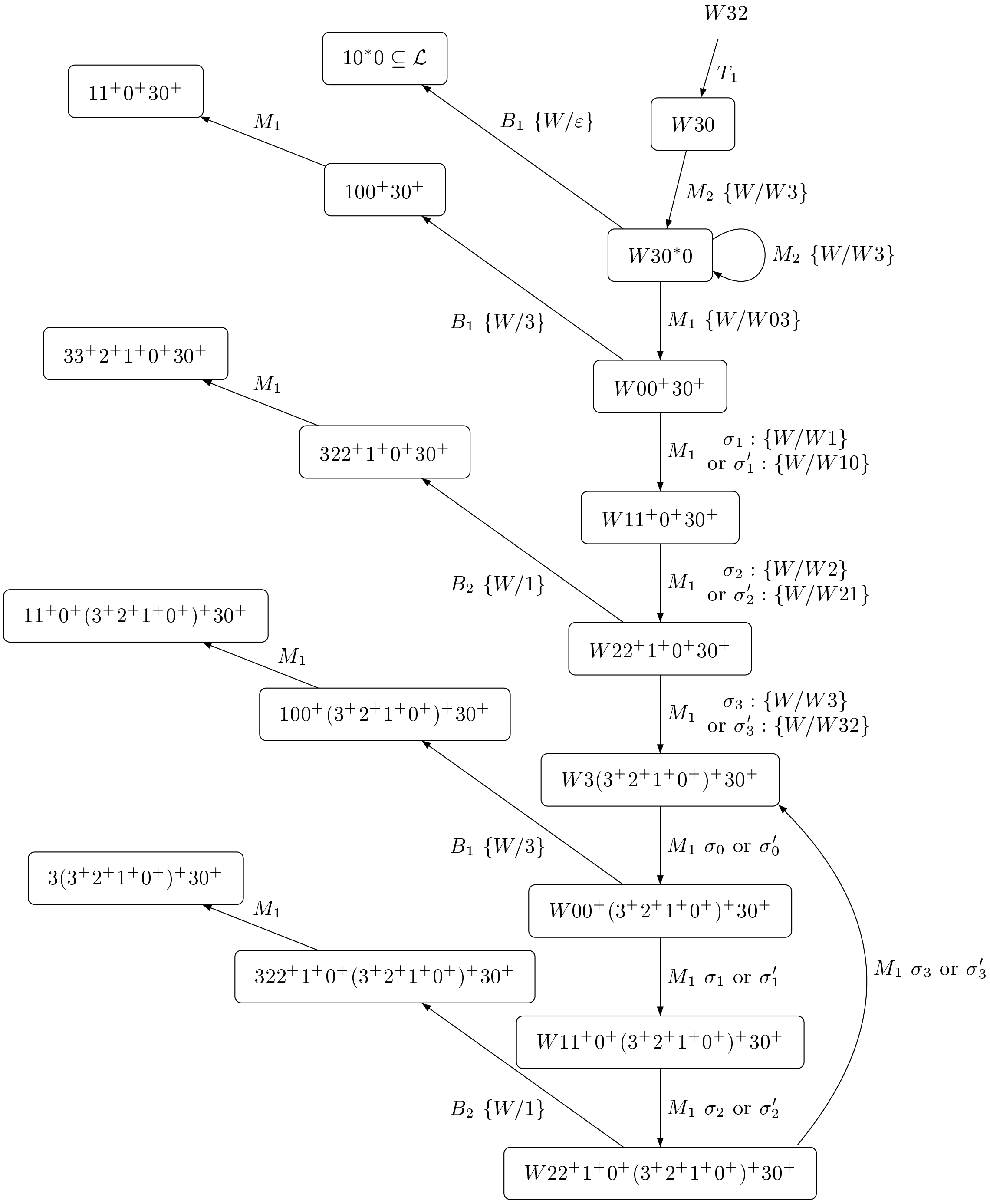

Fig. 3. First-order generation of chain for T_1

from the other top rule righthand side $\#W12\#$. To make the figure more readable, some details have been omitted:

- substitutions σ_0 and σ'_0 are respectively $\{W/W0\}$ and $\{W/W03\}$,
- the edges labelled by M_1 without a substitution correspond to $\sigma = id$,
- most schemes appearing in the figure are closed by (generalized) application of the rule M_1 with $\sigma = id$ and should have a loop which is not represented.

A similar graph can be constructed, starting from top chain T_2 instead of T_1. The important point is that, in both graphs, all the paths make use of the top rule a finite number of times (at most once). It follows by theorem 3 that the system is self-stabilizing.

5.3 Hoepman's ring orientation algorithm

We finally sketch out how our method adapts to uniform algorithms (algorithms without distinguished top, bottom or middle rules). We take the self-stabilizing ring orientation algorithm presented by Hoepman in [18], as an example. Our underlying assumptions about the existence of distinguished top rules Top and the termination of $\mathcal{S} - Top$, do not hold any longer in this context. The reasoning for proving convergence, is modified by using an assumption of fairness (alos used in [18]) instead: Any infinite sequence of rules modifies infinitely often the state of *every* machine of the ring.

Hoepman's algorithm is based on 16 rules applied to words with the alphabet $\Sigma = \{1_-, 1_+, 0_-, 0_+\}$ and uses the following notations:
$L_0 = \Sigma^* = \{0,1\}^*$ with $0 = \{0_+, 0_-\}$, $1 = \{1_+, 1_-\}$,
$L_1 = (O_1\ I_1)^*$ with $O_1 = 0 \cup \{0_+0_-0_+,\ \ 0_-0_+,\ \ 0_+0_-\}$ and $I_1 = 1 \cup \{1_+1_-1_+,\ \ 1_-1_+,\ \ 1_+1_-\}$,
$L_2 = (O_2\ I_2)^*$ with $O_2 = 0 \cup \{0_+0_-\}$ and $I_2 = 1 \cup \{1_+1_-\}$.

The corresponding rewriting system $\mathcal{S}$ transforms a subword pqr in $pq'r$, where q' is given by the following tables:

rule	p	q	r	q'
a	0	0	0	1_-
b	0	1	0	1_-
c	1	1	1	0_-
d	1	0	1	0_-

rule	p	q	r	q'
e	0_+	0_-	1_-	1_+
e'	1_-	0_-	0_+	1_+
f	1_+	1_-	0_-	0_+
f'	0_-	1_-	1_+	0_+

rule	p	q	r	q'
g	0_-	0_-	1	0_+
g'	1	0_-	0_-	0_+
h	0_+	0_+	1	0_-
h'	1	0_+	0_+	0_-

rule	p	q	r	q'
i	1_-	1_-	0	1_+
i'	0	1_-	1_-	1_+
j	1_+	1_+	0	1_-
j'	0	1_+	1_+	1_-

These rules can be applied at any position of the configuration. (Formally, every transformation of pqr into $pq'r$ corresponds to 3 rules: $\#XpqrY\# \to \#Xpq'rY\#$, $\#rXpq\# \to \#rXpq'\#$, $\#qrXp\# \to \#q'rXp\#$.) Hoepman proves that the system converges on L_2 in two steps: first, he exhibits a measure that strictly decreases, when applied to a ground configuration of L_0 unless this ground configuration belongs to subset L_1; second he exhibits another measure that strictly decreases, when applied to a ground configuration of L_1, unless it belongs to L_2. (He proves that the system converges from L_2 to a third set L_3, but this is beyond the scope of this presentation.) In contrast, our method gives a direct proof for the convergence to L_2, viewed here as set $\mathcal{L}$ of legitimate configurations, and does not appeal to any strictly decreasing mesure.

We preliminary transform Hoepman's system into a simpler set of rules. We first use a φ-refinement in order to disregard rules (a) and (c). Here measure φ counts the number of maximal subsequences of the same value (either 0 or 1), inside a word (assuming the leftmost and rightmost elements to be contiguous). For example, the measure φ for $\#000100W1100\#$ is 4. All the rules preserve this number, except rules (a) and (c), which strictly increase it by 2. Therefore, (a) and (c) can only be used a finite number of times (φ is bounded upward with N), and we can focus on infinite derivations via $\mathcal{S} - \{a, c\}$. The remaining rules are themselves merged into the simple new system $\mathcal{S}' = \{E, E', F, F'\}$ given by the table below.

rule	p	q	r	p'	q'	r'
E	0_+	0_-	1	0	1_+	1_-
E'	1	0_-	0_+	1_-	1_+	0
F	1_+	1_-	0	1	0_+	0_-
F'	0	1_-	1_+	0_-	0_+	1

Note that each transformation is now a shorthand for 4 transformations. For example E transforms $0_+0_-1_-$ or $0_+0_-1_+$ into $0_-1_+1_-$ or $0_+1_+1_-$. As before, every transformation of pqr into $p'q'r'$ corresponds to 3 rules:

$\#XpqrY\# \to \#Xp'q'r'Y\#$,
$\#rXpq\# \to \#r'Xp'q'\#$,
and $\#qrXp\# \to \#q'r'Xp'\#$,

depending on the position where it is applied.
It is easy to see that every sequence via $\mathcal{S} - \{a, c\}$ can be simulated by a sequence via $\mathcal{S}' = \{E, E', F, F'\}$.

In order to prove self-stabilization via $\mathcal{S}$, it is thus necessary and sufficient to prove that there is no infinite sequence of derivations via $\mathcal{S}'$ (apart from sequences ending at L_2). We thus focus on chains starting from a rule E (or E', F, F'). Since the system is uniform, we can actually focus on chains starting with rule E applied at an arbitrary position, e.g., the rightmost one. We thus take $E:\ 0_-1W0_+ \to 1_+1_-W0$ as a starting rule and we show that there is no infinite derivation from E by first-order rewriting via $\mathcal{S}'$ (regardless of sequences going to L_2). This is done in two steps: first we build the graph associated with E, then we explain why there is no infinite path in this graph (except those going to L_2).

This graph can be found in figure 4.

Only the edges corresponding to non trivial substitutions ($\sigma \neq id$) are represented. All nodes should have in addition a self-loop labeled by $\sigma = id$. For the sake of readability, the substitutions labeling the edges of the figure are also omitted. For example, consider the upmost node $(I_2O_2)^*1_+1_-W1_-1_+(O_2I_2)^*O_2$, with its three outgoing edges labeled respectively E, E', F':

- The substitution for rule E is $\{W/0_+0_-\}$, which gives $(I_2O_2)^*1_+1_-01_+1_-1_+(O_2I_2)^*O_2$, generalized as $\Sigma^*01_+1_-1_+\Sigma^*$.
- The substitution for rule E' is $\{W/0_-0_+\}$, which gives $(I_2O_2)^*1_+1_-1_+01_-1_+(O_2I_2)^*O_2$, generalized as $\Sigma^*1_+1_-1_+0\Sigma^*$.
- The substitution for rule F' is $\{W/W0\}$, which gives $(I_2O_2)^*1_+1_-W0_-0_+1(O_2I_2)^*O_2$, generalized as $(I_2O_2)^*1_+1_-W0_-0_+(I_2O_2)^*$.

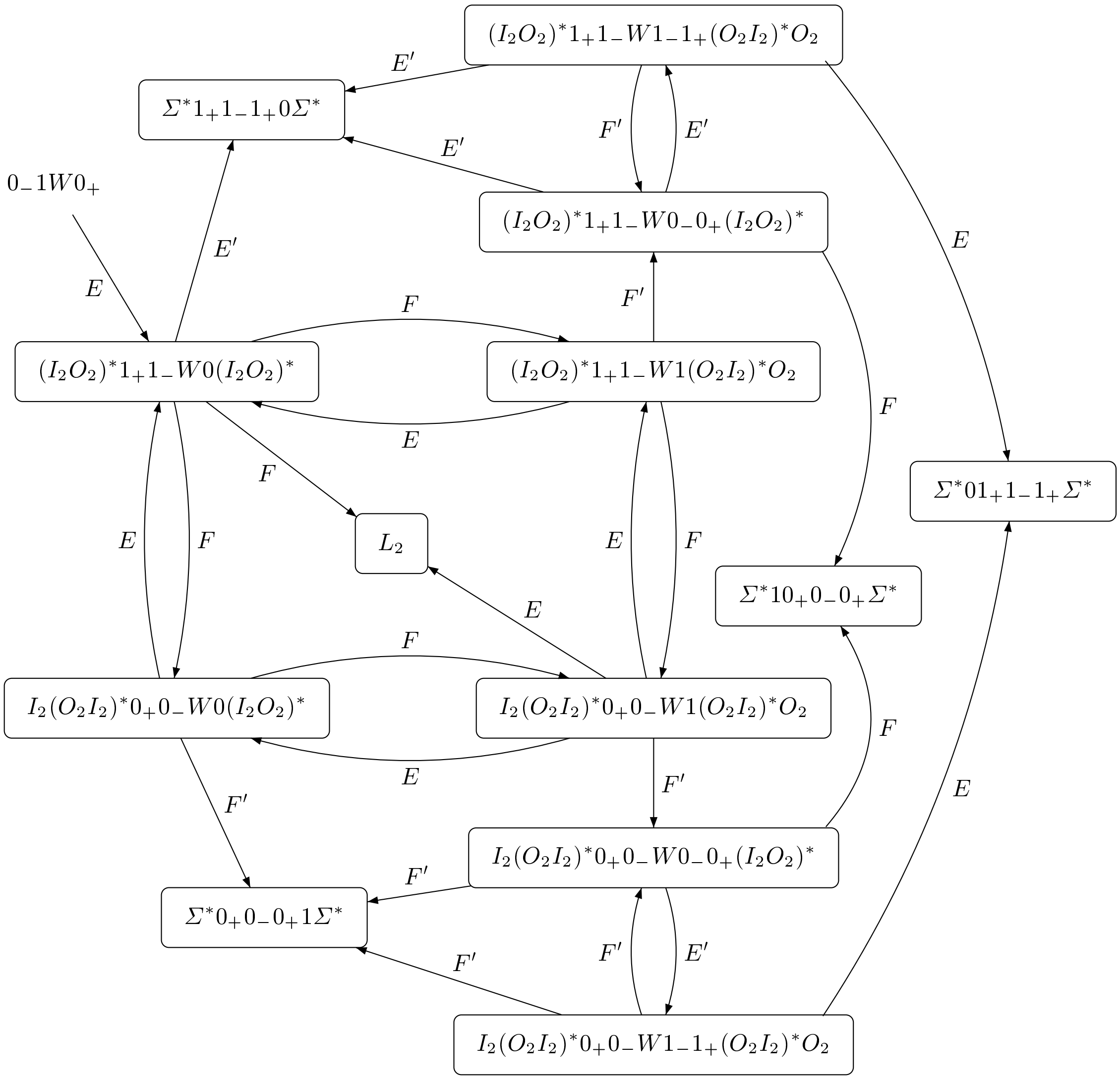

Fig. 4. First-order graph generation for E

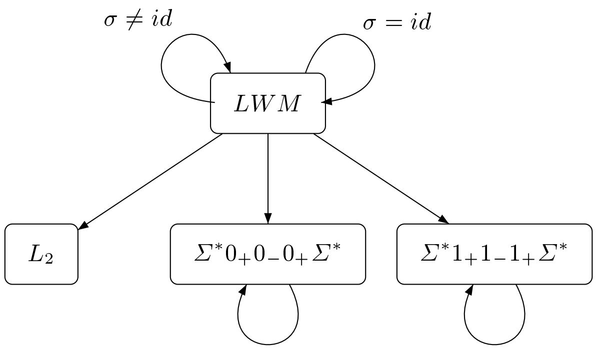

Fig. 5. A reduced form for the graph in Fig. 4

Also note that the two edges leading to L_2 (our set of legitimate configurations) are implicitly labeled by $\{W/\varepsilon\}$.

Considering this graph, we see that there are three kinds of nodes. The first one corresponds to a first order scheme LWM, the second one to the legitimate scheme L_2 and the third one to ground schemes containing either $0_+0_-0_+$ or $1_+1_-1_+$. This observation is summarized in figure 5.

A simple analysis of this synthetic form, leads us to conclude that no infinite path exists in this graph (except those passing in L_2), for the following reasons:

- There is no infinite loop in the nodes which contain patterns like $0_+0_-0_+$ (resp. $1_+1_-1_+$) because no rule can rewrite the central letter which is in contradiction with the fairness assumption.
- There is no infinite loop with a substitution $\sigma \neq id$ because X represents a finite word and then cannot be instanciated infinitely often.
- There is no infinite loop with label $\sigma = id$ because otherwise, the string represented by X could never be modified at its extremities which contradicts fairness.

Similar constructions and explanations hold for rules E', F, F'. This achieves our proof of self-stabilization.

6 Conclusion and perspectives

In contrast with methods relying on the existence of a strictly decreasing norm function [5,14,19,23], our technique 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 [14] and Beauquier-Debas's 3-state algorithm [5]. These examples are simple ones, which allow us to give a clear view of the procedure. Our procedure is inspired by Dershowitz's chain generation procedure [8], and proves convergence of self-stabilizing algorithms much in the same way than Dershowitz proves the termination of rewrite systems. Following Hoepman [18], we have enhanced the basic method by incorporating a process of generalization of words as regular languages, and defining rewriting over "schemes". The method is not fully automatic: we need in particular to infer by hand generic schemes of configurations from words produced recurrently throughout derivations.

The main differences with traditional proof methods of self-stabilization, which use strictly decreasing measures over ground configurations, come from:

1. focusing on derivations originating from *top*-configurations instead of derivations starting from arbitrary configurations.
2. reasoning with *1st-order variables*, and deriving new configurations through a restricted strategy of reduction (top chain generation) instead of considering all the possible ground configurations, and all their possible ground successors.
3. reasoning with *regular languages* (including 1st-order variables), and integrating a process of generalization.

From a mechanical point of view, the operation of minimal reduction over schemes can be implemented via the operation of *transducer* (see [24]; cf [4]). It is also possible to analyse further the constructed graph in order to extract a complexity upperbound for the convergence. In [4], further results are given, concerning a natural counterpart of Herman's compositionality result in our framework. We believe that our method adapts easily to the case of *uniform rings*, as sketched out in the case of Hoepman's ring-orientation protocol. We are currently investigating an extension of the method to algorithms running on arbitrary (non-ring) networks, that could use graph rewriting techniques as proposed in [25–27].

Acknowledgement

We would like to thank Nachum Dershowitz and two anonymous referees for their helpful comments.

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, B. Bérard, L. Fribourg and F. Magniette. *Proving Convergence of Self-Stabilizing Systems Using First-Order Rewriting and Regular Languages.* Research Report LSV-00-1, Lab. Specification and Verification, ENS de Cachan, Cachan, France, January 2000.
5. 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.
6. R.V. Book and F. Otto. *String-Rewriting Systems.* Springer-Verlag, 1993.
7. J.E. Burns and J. Pachl. "Uniform Self-Stabilizing Rings". *TOPLAS* 11:2, 1989, pp. 330–344.
8. N. Dershowitz. "Termination of Linear Rewriting Systems". *Proc. ICALP*, LNCS 115, Springer-Verlag, 1981, pp. 448-458.
9. N. Dershowitz and C. Hoot. "Topics in termination". *Proc. Rewriting Techniques and Applications*, LNCS 690, Springer-Verlag, 1993, pp. 198-212.
10. N. Dershowitz and J.-P. Jouannaud. "Rewrite Systems". *Handbook of Theoretical Computer Science*, vol. B, Elsevier - MIT Press, 1990, pp. 243-320.
11. E.W. Dijkstra. "Self-stabilizing systems in spite of distributed control". *Comm. ACM* 17:11, 1974, pp. 643-644.
12. E.W. Dijkstra. "A Belated Proof of Self-stabilization". *Distributed Computing* 1:1, 1986, pp. 5-6.
13. M. Fay. "First-order unification in an equational theory". *Proc. 4th Workshop on Automated Deduction*, Austin, Texas, 1979, pp. 161-167.
14. S. Ghosh. "An Alternative Solution to a Problem on Self-Stabilization". *ACM TOPLAS* 15:4, 1993, pp. 735-742.
15. M.G. Gouda. "The triumph and tribulation of system stabilization". *Proc. 9th WDAG*, LNCS 972, Springer-Verlag, 1995, pp. 1-18.
16. M.G. Gouda and T. Herman. "Adaptative programming". *IEEE Transactions on Software Engineering* 17, 1991, pp.911-921.
17. T. Herman. *Adaptativity through distributed convergence.* Ph.D. Thesis, University of Texas at Austin. 1991.
18. 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.
19. S.C. Hsu and S.T. Huang. "A self-stabilizing algorithm for maximal matching". *Information Processing Letters* 43:2, 1992, pp. 77-81.

20. J.-M. Hullot. "Canonical Forms and Unification". *Proc. 5th Conf. on Automated Deduction*, LNCS, Springer-Verlag, 1980, pp. 318-334.
21. J. Jaffar. "Minimal and Complete Word Unification". *J. ACM* 37:1, 1990, pp. 47-85.
22. S. Katz and K.J. Perry. "Self-stabilizing extensions for message-passing systems". *Distributed Computing* 7, 1993, pp. 17-26.
23. J.L.W. Kessels. "An Exercise in proving self-stabilization with a variant function". *Information and Processing Letters* 29, 1988, pp. 39-42.
24. 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.
25. I. Litovski and Y. Métivier. "Computing with Graph Rewriting Systems with Priorities". *Theoretical Computer Science* 115:2, 1993, pp. 191-224.
26. I. Litovski, Y. Métivier and E. Sopena. "Different Local Controls for Graph Relabeling Systems". *Mathematical Systems Theory* 28:1, 1995, pp. 41-65.
27. 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.
28. G.S. Makanin. "The problem of solvability of equations in a free semigroup". *Matematiceskij Sbornik* 103, 1977, pp. 147-236.
29. M. Schneider. "Self-Stabilization". *ACM Computing Surveys* 25:1, 1993.
30. J.R. Slagle. "Automated Theorem-Proving for Theories with Simplifiers, Commutativity, and Associativity." *J. ACM* 21:4, 1974, pp. 622-642.
31. G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 1994.
32. 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.