Self-stabilization with Path Algebra

Bertrand Ducourthial
UMR CNRS 6599 HEUDIASYC
Université de Technologie de Compiègne
Bertrand.Ducourthial@utc.fr

Sébastien Tixeuil
UMR CNRS 8623 LRI
Université de Paris-Sud
tixeuil@lri.fr

June 2000

Abstract

We have proposed a simple parametric algorithm to solve distributed static tasks. When parametrized with specific local functions, this algorithm can solve various classical tasks while supporting transient faults. We extend previous results using max-plus algebra and asynchronous iteration towards a matrix modeling of the system.

Summary

- Self-stabilization
- Related works

- Parametric algorithm
- r-operators
- Operators for silent tasks
- Operators for self-stabilizing tasks

- Matrix representation
- Synchronous demon
- Distributed demon
- Fully distributed demon
- Max-plus matrix proof

- New applications

- Conclusion

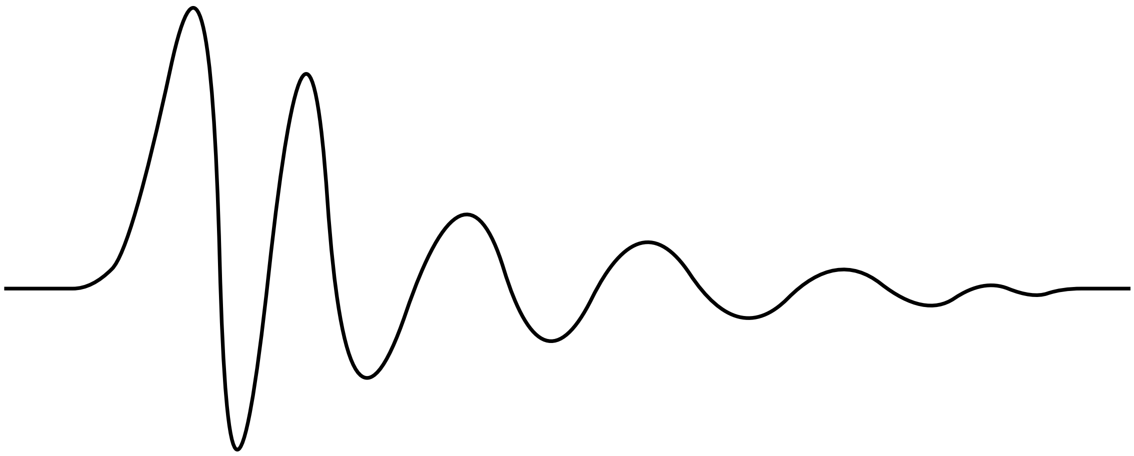
Self-stabilization

Transient faults

- fault $\rightsquigarrow$ modification of data
- registers, RAM, but not in ROM
- not for the code ($e.g.$, electronic display)

Self-stabilizing

- once fault cease, the system recovers in a finite time a safe state [Dijkstra 74, Schneider 93]



Some related work

- Global dynamic condition
  - Arora, Attie, Evangelist, Gouda (1993):
    Convergence of Iteration Systems

- Global static condition
  - Herman (1991):
    Adaptativity through distributed convergence
  - Tel (1991):
    Topics in Distributed Algorithms, chap. 4

- Global static condition for metric trees
  - Gouda, Schneider (1995):
    Maximum flow routing

  - Gouda, Schneider (1999):
    Stabilization of maximal metric trees

Related work

Self-stabilization

- undirected graphs [Dolev Israeli Moran 93, Afek et all 90]
- directed graph, stongly connected [Alstein et all 94]
  $\leadsto$ virtual well known topology (tree) [Alima et all 98]

Silent tasks

- communications between processors are fixed from some point of the execution [Dolev Gouda Schneider 96]
- communication with registers $\leadsto$ their content do not change after some point in the execution

Self-stabilizing silent tasks

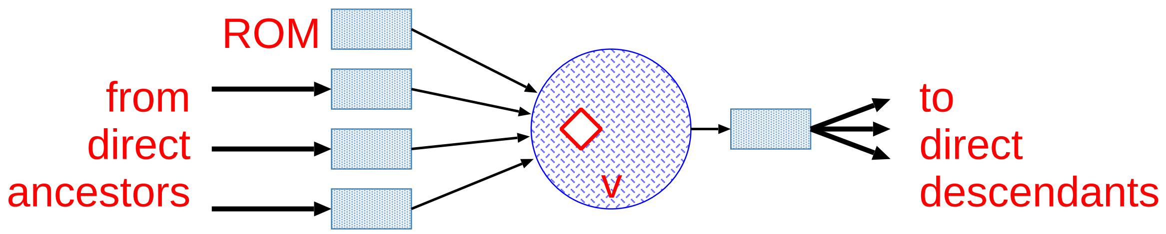
- directed graphs, not strongly connected
- but without cycles $\leadsto$ DAG [Bourgon et all 95]

Our work

- self-stabilizing silent task on any directed graph
- provided there exists a specific operator for instanciated the algorithm

Parametrized algorithm

- Node



- Local computation
  - binary operator: $\text{reg}_{\text{out}} \leftarrow \text{reg}^0_{\text{in}} \diamond \cdots \diamond \text{reg}^{\delta^-_v}_{\text{in}}$
  - n-ary operator: $\text{reg}_{\text{out}} \leftarrow \diamond \left(\text{reg}^0_{\text{in}}, \ldots, \text{reg}^{\delta^-_v}_{\text{in}}\right)$

- Local parametric algorithm
  - Copying rule: if data of one ancestor is not identical to the copy in the private register, then the register is updated
  - Computing rule: if the data in the outcoming register is not equal to the result of the local computation, then the register is updated

- What sort of operators to obtain silent/self-stabilizing tasks ?

r-operators.

s-operator $\oplus$ (infimum)

associative	$(x \oplus y) \oplus z = x \oplus (y \oplus z)$
commutative	$x \oplus y = y \oplus x$
idempotent	$x \oplus x = x$
identity element	$x \oplus e_{\oplus} = x$

r-operator

$\triangleleft$ is an r-operator on $\mathbb{S}$ if their exists a bijective mapping $\mathsf{r} : \mathbb{S} \to \mathbb{S}$ such that $\triangleleft$ is :

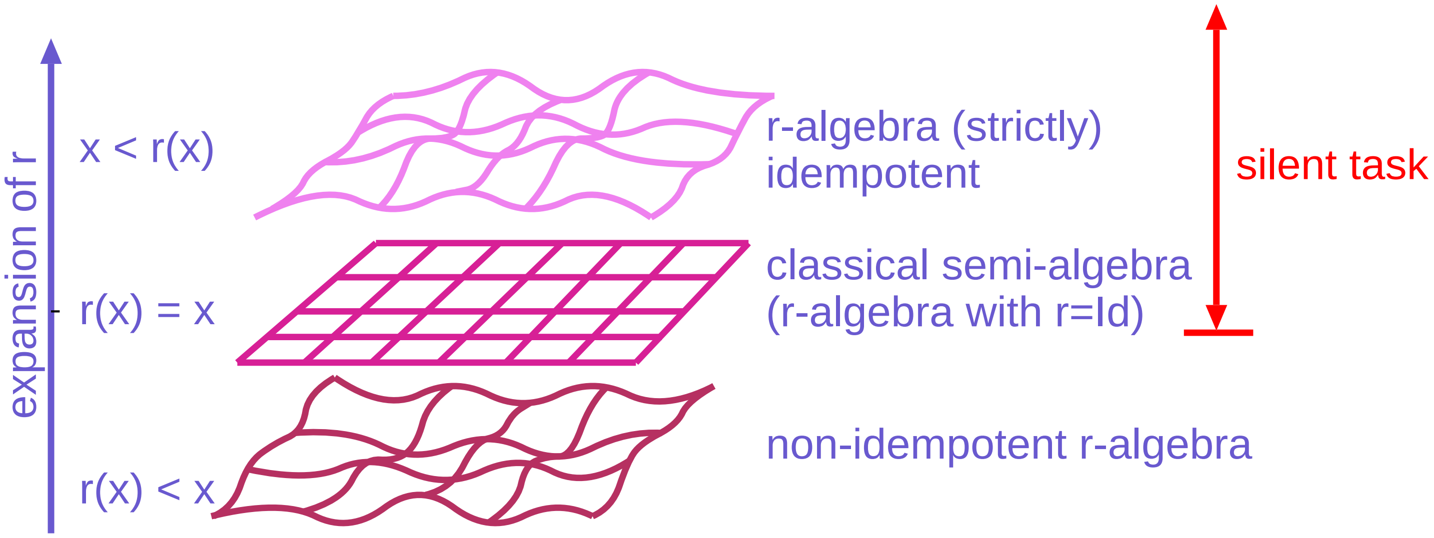
r-associative	$(x \triangleleft y) \triangleleft \mathsf{r}(z) = x \triangleleft (y \triangleleft z)$
r-commutative	$\mathsf{r}(x) \triangleleft y = \mathsf{r}(y) \triangleleft x$
r-idempotent	$\mathsf{r}(x) \triangleleft x = \mathsf{r}(x)$
right identity elt	$x \triangleleft e_{\triangleleft} = x$

Examples

- s-operators: min, max, and, or, $\cup$, $\cap$, $\mathbf{lcm}$, $\mathbf{gcd}$, ...
- r-operators: $\mathrm{minc}(x, y) = \min(x, y + 1)$

$$\mathrm{minc}_w(x_0, \ldots, x_n) = \min(x_0, x_1 + w_1, \ldots, x_n + w_n)$$
$$\mathrm{maxmul}_\pi(x_0, \ldots, x_n) = \max(x_0, x_1 \times \pi_1, \ldots, x_n \times \pi_n)$$
$$\mathrm{lexicat}\left((a, b, c), (a', b', c')\right) = (a, b, c) \oplus (a', b', c', v)$$

Operators for silent tasks.

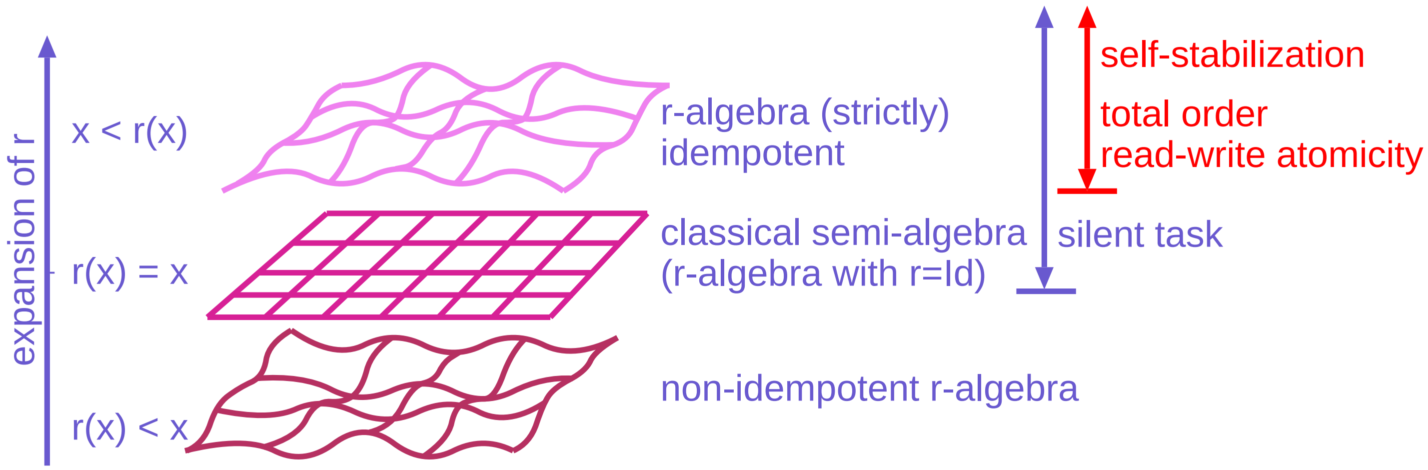


$$x \triangleleft y = x \oplus \mathsf{r}(y)$$

- If the algorithm is instanciated with an idempotent r-operator $\triangleleft$, then the distributed algorithm is silent. [SIROCCO98]

- read-write atomicity

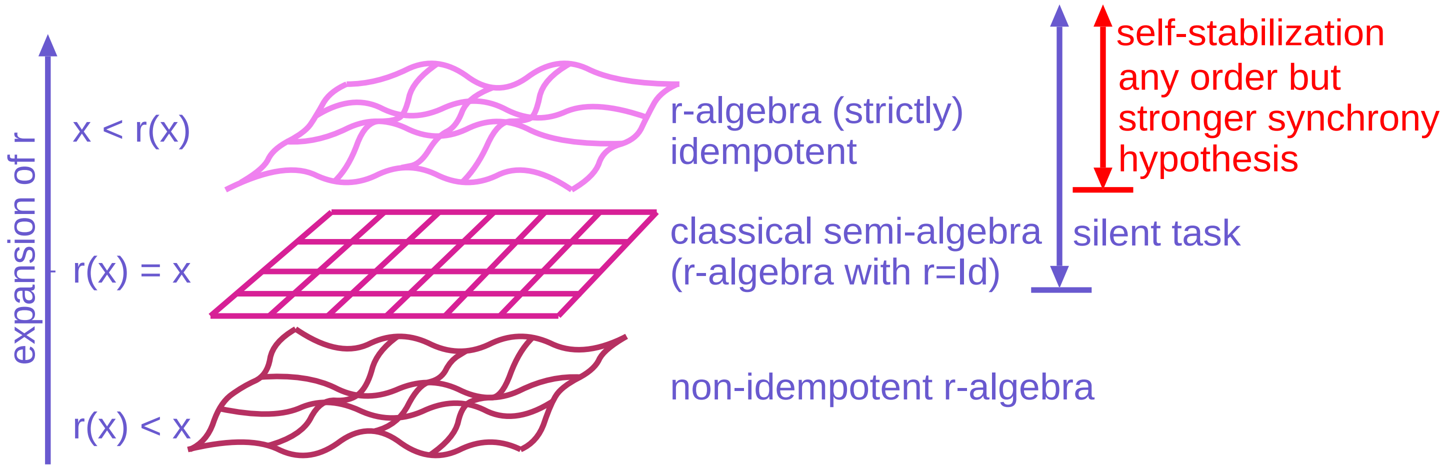
Operators for self-stabilizing tasks (1).



$$x \lhd y = x \oplus \mathsf{r}(y)$$

- If the algorithm is instanciated with a strictly idempotent r-operator $\lhd$ based on a total ordering relation $\oplus$, then the algorithm is self-stabilizing. [OPODIS98]
- Strict idempotency is necessary.
- Read-write atomicity
- $\oplus$ defines a total order on $\mathbb{S}$

Operators for self-stabilizing tasks (2).



$$x \lhd y = x \oplus \mathsf{r}(y)$$

- Using a stronger synchrony hypothesis than read-write atomicity, the algorithm is self-stabilizing with any order. [SIROCCO00]

- Strict idempotency is necessary.

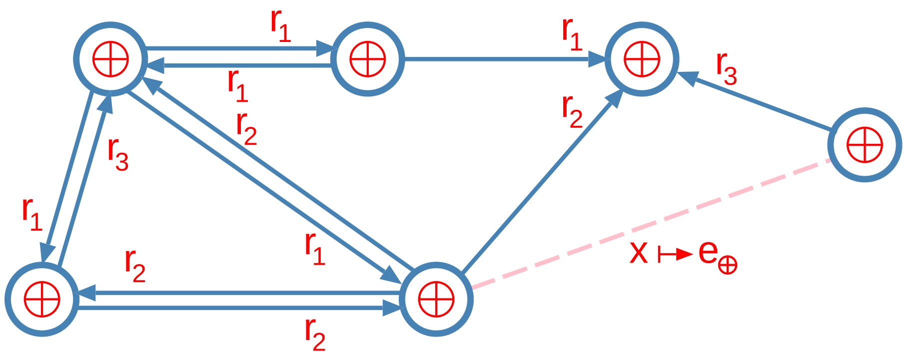
-

total order of $\oplus$	read-write atomicity
any order of $\oplus$	stronger synchrony hypothesis

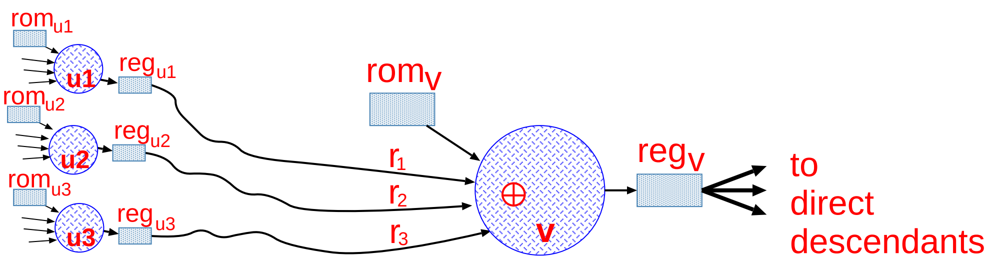
- Proof by max-plus algebra and asynchronous iteration (matrix representation)

Matrix representation.

- $N \times N$ matrix $\mathbf{M}$ of mappings $\mathsf{r}_i : \mathbb{S} \to \mathbb{S}$



- Vectors of N components REG and ROM



$\leadsto \neq$ read-write atomicity

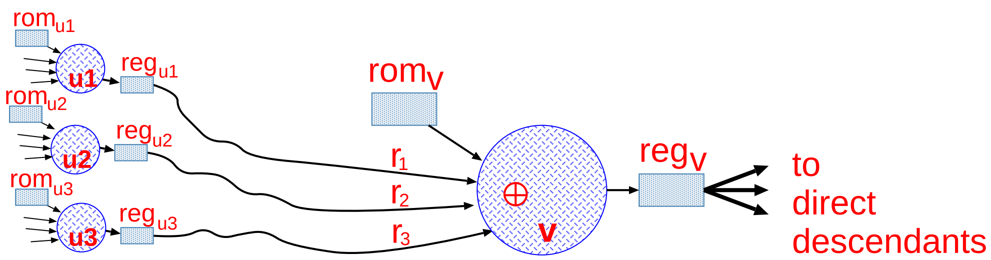
- System modeling

$$\mathbf{M} = \begin{pmatrix} \mathsf{r}_i & \dots & \mathsf{r}_j \\ \vdots & & \vdots \\ \mathsf{r}_p & \dots & \mathsf{r}_q \end{pmatrix} \quad \mathtt{REG} = \begin{pmatrix} \mathsf{reg}_u \\ \vdots \\ \mathsf{reg}_w \end{pmatrix} \quad \mathtt{ROM} = \begin{pmatrix} \mathsf{rom}_u \\ \vdots \\ \mathsf{rom}_w \end{pmatrix}$$

$\leadsto$ computation modeling ?

Synchronous demon.

- Local computation



$$\mathsf{reg}_v = \mathsf{rom}_v \oplus \mathsf{r}_1\left(\mathsf{reg}_{u_1}\right) \oplus \cdots \oplus \mathsf{r}_{\delta_v^-}\left(\mathsf{reg}_{u_{\delta_v^-}}\right)$$

- Global computation
synchronous demon $\rightsquigarrow$ all processors read their inputs and compute their result simultaneously

$$\mathtt{REG}_{n+1} = \mathbf{M} \otimes \mathtt{REG}_n \;\oplus\; \mathtt{ROM}$$

Distributed demon.

- Synchronous demon
  all processors read their inputs and compute their result simultaneously.
  $\mathtt{REG}$: vector of results at step n

$$\mathtt{REG}_{n+1} = \mathbf{M} \otimes \mathtt{REG}_n \quad \oplus \quad \mathtt{ROM}$$

- Distributed demon.
  a subset $\mathcal{J}$ of processors read their inputs and compute their result simultaneously.
  $\mathcal{J}_n$: subset of activated processors at step n

$$\mathtt{REG}_{n+1}[i] = \begin{cases} \mathtt{REG}_n[i] & \text{if } i \notin \mathcal{J}_n \\ \mathbf{M} \otimes \mathtt{REG}_n \quad \oplus \quad \mathtt{ROM} \;\big|_i & \text{if } i \in \mathcal{J}_n \end{cases}$$

$\leadsto$ asynchronous iteration

Fully distributed demon.

- Distributed demon.
  a subset $\mathcal{J}$ of processors read their inputs and compute their result simultaneously.
  $\mathcal{J}_n$: subset of activated processors at step n

$$\mathtt{REG}_{n+1}[i] = \begin{cases} \mathtt{REG}_n[i] & \text{if } i \notin \mathcal{J}_n \\ \mathbf{M} \otimes \mathtt{REG}_n \quad \oplus \quad \mathtt{ROM} \;\Big|_i & \text{if } i \in \mathcal{J}_n \end{cases}$$

- Fully distributed demon.
  a subset $\mathcal{J}$ of processors read their inputs and compute their result simultaneously but their writting may be delayed until after following demon activations.
  $\mathtt{REG}_{D_n[i]}[j]$: result of node j has been produced at step $D_n[i]$ instead of n

$$\mathtt{REG}_{n+1}[i] =$$
$$\begin{cases} \mathtt{REG}_n[i] & \text{if } i \notin \mathcal{J}_n \\ \mathbf{M} \otimes (\mathtt{REG}_{D_n[1]}[1], \ldots, \mathtt{REG}_{D_n[N]}[N])^{\mathsf{t}} \oplus \mathtt{ROM} \;\Big|_i & \text{if } i \in \mathcal{J}_n \end{cases}$$

$\rightsquigarrow$ asynchronous iteration with delay

Max-plus matrix proof (1).

- Asynchronous iteration

$$\begin{array}{rlcl} \mathsf{F}: & \mathbb{S}^N & \to & \mathbb{S}^N \\ & \mathtt{REG}_{n+1} & \mapsto & \mathsf{F}\left(\mathtt{REG}_n\right) \end{array}$$

- Conditions for convergence [Üresin and Dubois]

1. F is closed on $\mathbb{S}^N$
2. the synchronous iterations converge and $\mathtt{REG}_{n+1} \preceq_{\oplus} \mathtt{REG}_n$
3. F is monotonous on $\mathbb{S}^N$: $\mathtt{X} \preceq_{\oplus} \mathtt{Y} \Rightarrow \mathsf{F}(\mathtt{X}) \preceq_{\oplus} \mathsf{F}(\mathtt{Y})$

$\leadsto$ convergence of synchronous iteration ?

Max-plus matrix proof (2).

Synchronous iteration

- Synchronous iteration: $\mathtt{X}_k = \mathbf{M} \otimes \mathtt{X}_{k-1} \oplus \mathtt{B}$.
- Rewritten as: $\mathtt{X}_k = \mathbf{M}^k \otimes \mathtt{X}_0 \oplus \mathbf{M}^{(k-1)} \otimes \mathtt{B}$
- Where $\mathbf{M}^{(k)} = \mathbf{E}_\otimes \oplus \mathbf{M} \oplus \cdots \oplus \mathbf{M}^k$

First term

- Let $\mathsf{r}_\perp : \mathbb{S} \to \mathbb{S}$
  defined by $\forall x \in \mathbb{S}, \mathsf{r}_\perp(x) = \bigoplus_{i \in \{1,\ldots,|E|\}} \mathsf{r}_i(x)$
- Let matrix $\mathbf{R}_\perp$ defined by $\mathbf{R}_\perp[i][j] = \mathsf{r}_\perp$.
- $\mathbf{R}_\perp \preceq_\oplus \mathbf{M}$.
- Strict idempotency $\Rightarrow \lim_{k\to+\infty} \mathbf{R}_\perp^k = \mathbf{E}_\oplus$.
- $\leadsto \lim_{k\to+\infty} \mathbf{M}^k = \mathbf{E}_\oplus$.

Second term

- Strict idempotency $\Rightarrow \mathsf{e}_\otimes \oplus \mathsf{r}_i = \mathsf{e}_\otimes$ for all $i \in \{1, \ldots, |E|\}$
- Then, there is no absorbing circuit in the network.
- $\leadsto \lim_{k\to+\infty} \mathbf{M}^{(k-1)} = \mathbf{M}^*$. [Gondran79].

Conclusion

$$\forall\, \mathtt{X}_0 \in \mathbb{S}^n, \quad \exists k_0 \in \mathbb{N} \quad \text{such that,} \quad \text{for } k \geq k_0,$$
$$\mathtt{X}_k \quad = \quad \mathbf{M}^k \otimes \mathtt{X}_0 \;\oplus\; \mathbf{M}^{(k-1)} \otimes \mathtt{B} \quad = \quad \mathbf{M}^* \otimes \mathtt{B}$$

$\leadsto$ the synchronous iteration $\mathtt{X}_{k+1} = \mathbf{M} \otimes \mathtt{X}_k \oplus \mathtt{B}$ converges for *any initial guess* $\mathtt{X}_0$.

New application.

- Ordered ancestor list

Conclusion.

- ## Previous results

  local parametric algorithm
  self-stabilizing when instanciated with strictly-idempotent r-operators based on total ordering relation, under read-write atomicity

- ## Extension

  total order is not needed under fully distributed demon

- ## Proof

  r-operators $+$ fully distributed demon $\rightsquigarrow$ max-plus matrix modeling