r-semi-groups: an algebra for computing on networks

Bertrand Ducourthial*

May 31, 2006

Abstract

Computing on a network can be seen as performing local computations on the nodes of the network (computing units, processors or computers, depending on the context) interlaced by communication between nodes. Local algorithms can be assimilated to operators, that compute a new result (output) with the received inputs (outputs of ancestor nodes in the network as well as local inputs). Such a modeling is quite general and can be used both in parallel and distributed computing. It allows for instance to study the termination or the stability against failures of the network computation by means of algebraic properties of the operators. This approach gives generic results that can be applied on a wide range of applications. Moreover, checking conditions on the local operator is a convenient way for ensuring global properties.

In this report, the algebraic properties of the operators are studied regarding their applications for network computations. Idempotent Abelian semi-groups (such as $(\mathbb{N}, \min)$) have already been considered for such computations in networks. We propose a generalization of these semi-groups, called r-semi-groups.

Several important applications of the r-operators (laws of the r-semi-groups) have already been studied both in parallel and distributed computing. The r-operators appear as a coherent framework for computing in a network, with practical applications. This paper presents a general view of this algebra and its applications.

*Lab. Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne (ducourth@hds.utc.fr)

Contents

1 Introduction **3**
1.1 Definitions . . . 3
1.2 Problem statement . . . 4

2 About idempotent Abelian semi-groups **6**

3 Some forms of idempotency **7**
3.1 Sequence idempotency . . . 8
3.2 Order idempotency . . . 9
3.3 Summary . . . 10

4 Weak left cancellation **10**

5 Rank properties **12**
5.1 Rank commutativity . . . 12
5.1.1 Rank k commutativity . . . 12
5.1.2 Rank 2 commutativity . . . 12
5.1.3 Usefulness of rank 2 commutativity . . . 13
5.1.4 Summary . . . 14
5.2 Rank idempotency . . . 14
5.2.1 Rank k idempotency . . . 14
5.2.2 Rank 2 idempotency . . . 15
5.2.3 Usefulness of rank 2 idempotency . . . 15
5.2.4 Summary . . . 16
5.3 Rank associativity . . . 16
5.4 Summary . . . 17

6 r-properties **17**
6.1 r-associativity . . . 17
6.2 r-commutativity . . . 19
6.3 r-idempotency . . . 21

7 r-operators **22**
7.1 r-semi-group . . . 22
7.2 Surjective mapping versus endormorphism . . . 24
7.3 Order relation . . . 25
7.4 Induced semi-group . . . 25

8 Applications of the r-operators **26**
8.1 Related work . . . 26
8.2 Applications in parallel computing . . . 28
8.3 Application to distributed computing . . . 29
8.4 Using r-operators . . . 31
8.5 Designing new r-operators . . . 32

9 Conclusion **33**

1 Introduction

In this section, we first define the problem we adress in this paper. Next, we explain its interest and our motivations in this study. Finally, we summarize the related works.

1.1 Definitions

Network. We consider a network modeled by a directed graph $G(V, E)$, composed of nodes (vertices $v \in V$) and unidirectional communication links (edges $(u, v) \in E$), such that each node owns a *private data*, some *inputs*, an *output* and an *operator* (see Figure 1). The topology of the network is fixed but unknown.

The private data, inputs and outputs represent some kind of memory cells. The private data is fixed, while the inputs and outputs are updated from time to time. Reads and writes are atomic on these variables, meaning that a variable cannot be read (resp. write) if it is currently writing (resp. reading).

Each edge (u, v) connects the output of u to the input of v, and is used to update the input of v by the output of u.

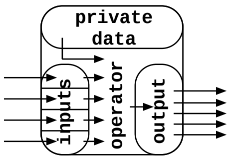

Figure 1: A node.

Local computation. A *local computation* is performed by a node v, and consists in updating its output by applying its operator (denoted by $\diamond$) on its private data and its inputs as follows (Figure 1):

$$\text{output} \leftarrow \text{private data} \diamond \text{first input} \diamond \cdots \diamond \text{last input} \tag{1}$$

We assume the following hypothesis in order the local computation is well defined:

Hypothesis 1 *All the data in the network belong to the set $\mathbb{S}$. Moreover, all the operators used by the nodes are internal composition law defined on $\mathbb{S}$ ($\forall x, y \in \mathbb{S}, x \diamond y \in \mathbb{S}$).*

Initialization. Since a local computation can be performed at any time, without any consideration of the communications, a node could perform a local computation before having received any value from some of its ancestors. We then admit the following hypothesis:

Hypothesis 2 *The operator* $\diamond$ *admits a* right identity element.

Supposing that all the inputs of a node v are initialized by a right identity element $e_\diamond$ of the local operator $\diamond$ used by the node, the output obtained by a local computation is equal to the private data x:

$$x = x \diamond e_\diamond \diamond \cdots \diamond e_\diamond$$

Hence, each input is initialized by a right identity of the local operator, and each output is initialized by the private data of the node. Note however that, as we will see, some operators tolerate any kind of initialization.

Global computation. A *global computation* begins after each node has been initialized as explained above, and may or not terminate in finite time. When it terminates on a deterministic result, such a computation defines a *network computing primitive*, that takes as inputs a network topology modeled by a directed graph, one private data per node, and one operator per node. It gives as result one output per node (Figure 2).

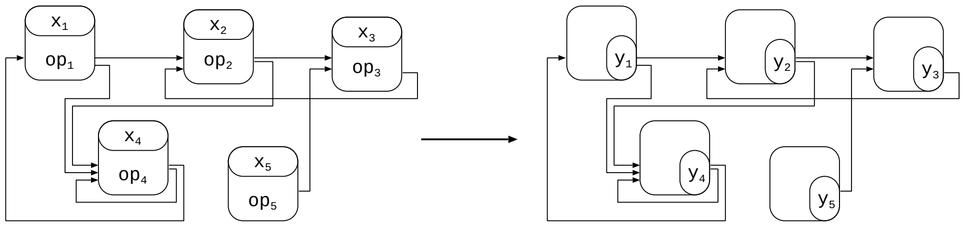

Figure 2: When it is defined, a global computation gives one result per node (y_i) built with the operators (op_i) and the private data (x_i).

One may consider that there is no kind of synchronization between nodes, nor between local computations and communications on the adjacent edges. In particular, when it is performed, a local computation may take into account only received inputs (and not the last outputs produced by its ancestors). Moreover, the output of a node may be updated by a local computation while the previous output has not yet been sent to the descendants in the network. This in fact depends on the context of applications. The proofs have essentially be done in a distributed system context, with several kind of synchronization hypotheses [9, 14, 15, 7] (see Section 8.3).

1.2 Problem statement

In this paper, we investigate the existence of this network computing primitive, which depends on the graph G, the private data of the nodes and the operators used by the nodes.

When the directed graph G has no circuit, the outputs of the *sources* are always equal to their private data. The output of a source's successor stabilizes on a value equal to the result of an expression composed of private data and operators of the ancestor sources of the node itself. All the node outputs, from the sources to the sinks, stabilize similarly. However the result of each node does depend on the underlying network (which is responsible of the form of the output expressions) while several different networks can be modeled by the same graph (see Figure 3).

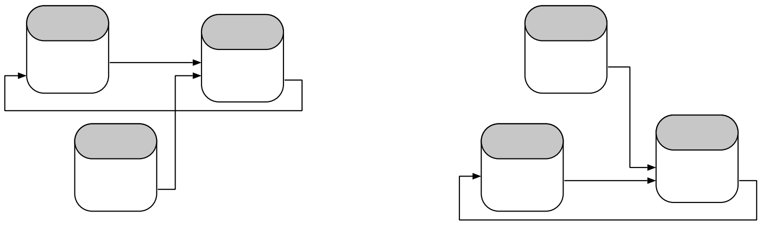

Figure 3: Two representations of the same graph.

Requirement 1 *The operators should not be aware of the network wiring, in order to obtain a result that depends on the graph topology (and not on the local neighborhood numbering).*

Note that, while this requirement is fulfilled by associative and commutative operators, other operators with weaker properties can also be used.

When the directed graph G has some circuits, the outputs could never stabilize. This is the case on the loop graph (see Figure 4) with the addition on the integers $\mathbb{N}$.

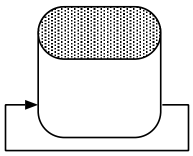

Figure 4: Loop. There is no termination in $(\mathbb{N}, +)$.

Requirement 2 *The operators should not be aware of the circuits.*

Note that, while this requirement is fulfilled by idempotent operators, other operators with weaker properties can also be used.

The operators of the idempotent semi-groups (such as min on the integer for instance) fulfill the requirements 1 and 2. We will see that other operators could be used instead, allowing more interesting applications. Indeed, the minimum produces the same result on all the nodes belonging to the same circuit. More complex results could be expected, for more complex applications.

2 About idempotent Abelian semi-groups

In this section, we give some definitions on idempotent Abelian semi-groups. Figure 5 gives a summary.

A *magma* $(\mathbb{S}, \diamond)$ is a set $\mathbb{S}$ endowed by an internal composition law $\diamond$ $(\forall x, y \in \mathbb{S}, x \diamond y \in \mathbb{S})$. A *monoid* is a magma with an associative law $(\forall x, y, z \in \mathbb{S}, x \diamond (y \diamond z) = (x \diamond y) \diamond z)$ [6]. An element $x \in \mathbb{S}$ is *left* (resp. *right*) *cancellative* if $\forall y, z \in \mathbb{S}$, $(x \diamond y = x \diamond z) \Rightarrow (y = z)$ (resp. $(y \diamond x = z \diamond x) \Rightarrow (y = z)$). It is cancellative (also called *regular*) if it is both left- and right-cancellative. If all the elements are cancellative, the monoid is said *cancellative* (or *free*). A *right* (resp. *left*) *identity element* $e_\diamond$ is defined by: $\forall x \in \mathbb{S}$, $x \diamond e_\diamond = x$ (resp. $e_\diamond \diamond x = x$). An *identity element* is a left and right identity element. A monoid with an identity element is called *semi-group*. It is a group without the notion of inverse elements. If the law $\diamond$ is commutative $(\forall x, y \in \mathbb{S}, x \diamond y = y \diamond x)$, the semi-group is said *commutative* or *Abelian*. If it is idempotent $(\forall x \in \mathbb{S}, x \diamond x = x)$, the semi-group is idempotent.

Any semi-group admits a pre-order relation $\preccurlyeq_\diamond$ (*i.e.*, a reflexive and transitive binary relation) defined by $(x \preccurlyeq_\diamond y) \equiv (\exists z \in \mathbb{S}, x = y \diamond z)$. We have $x \preccurlyeq_\diamond e_\diamond$ for any $x \in \mathbb{S}$ because $x = e_\diamond \diamond x$. When the relation $\preccurlyeq_\diamond$ is anti-symmetric (*i.e.*, $x \preccurlyeq_\diamond y \wedge y \preccurlyeq_\diamond x \Rightarrow x = y$), it is a partial order relation, and the semi-group is said *naturally ordered* by $\preccurlyeq_\diamond$.

A *null-sum* (*null-equation*) is of the form: $x \diamond y = e_\diamond$. A naturally ordered semi-group does not admit any null-sum. Indeed, $x \diamond y = e_\diamond$ implies $e_\diamond \preccurlyeq_\diamond x$, and then $e_\diamond = x$ (because $x \preccurlyeq_\diamond e_\diamond$). Then $e_\diamond \diamond y = e_\diamond$, and $x = y = e_\diamond$.

Moreover, a cancellative monoid without any null-equation is naturally ordered. Indeed, suppose that $x \preccurlyeq_\diamond y$ and $y \preccurlyeq_\diamond x$. Then it exists z and t in $\mathbb{S}$ such that $x = y \diamond z$ and $y = x \diamond t$, which gives $x = x \diamond t \diamond z$. Since the monoid is cancellative, we have $e_\diamond = t \diamond z$. Since there is no null-sum, we have $t = z = e_\diamond$. Thus $x = y$ and the pre-order relation is anti-symmetric. The monoid is then naturally ordered.

An idempotent semi-group naturally ordered by $\preccurlyeq_\diamond$ satisfies $x \preccurlyeq_\diamond y \equiv x = y \diamond x$. Indeed, if $x \preccurlyeq_\diamond y$, there exists $z \in \mathbb{S}$ such that $x = y \diamond z$. In this case, $y \diamond x = y \diamond y \diamond z$. By idempotency, we obtain $y \diamond x = y \diamond z$, and finally $y \diamond x = x$. The element $y \diamond x$ is the largest element of $\mathbb{S}$ (in sense of $\preccurlyeq_\diamond$) which is both smaller than x and y.

Since a semi-group owns an identity element, if it is both cancellative and idempotent, it owns a single element. Indeed, $x \diamond e_\diamond = x = x \diamond x$ gives $x = e_\diamond$ for any $x \in \mathbb{S}$. Hence, there is a schism between cancellative semi-groups (and groups) and idempotent semi-group.

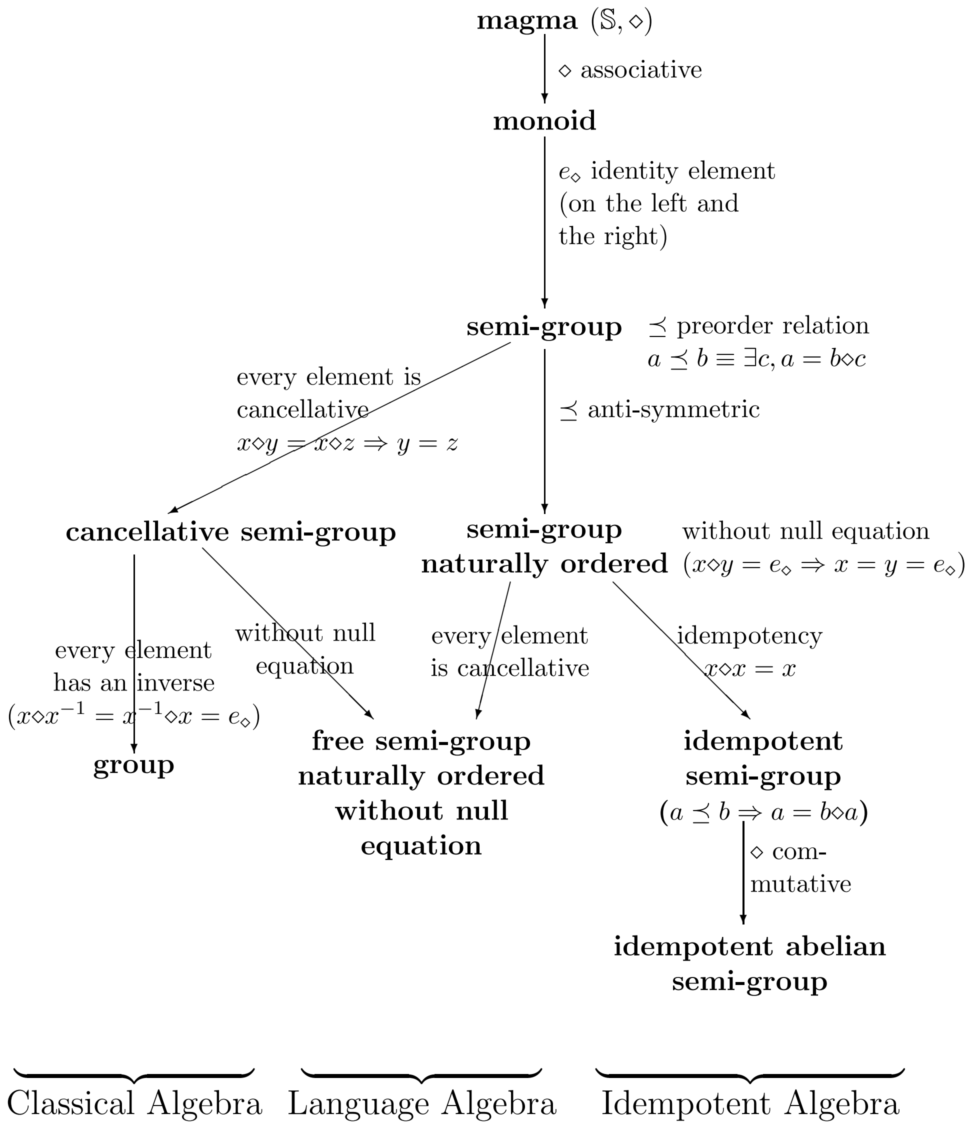

Figure 5: Summary of algebraic structures with one law.

3 Some forms of idempotency

By considering the network of Figure 4, it is obvious that some kind of idempotency is necessary to ensure the convergence in finite time of a global computation with any private data. And as already said, the idempotent semi-group $(\mathbb{N}, \min)$ fulfils the requirements 1 and 2. In this section, we introduce some weaker forms of idempotency.

3.1 Sequence idempotency

In the loop graph, when the private data of the node is x, the successive outputs are: x, $x \diamond x$, $x \diamond (x \diamond x)$, $x \diamond (x \diamond (x \diamond x))$... Let define the sequence $(x^{(n)})$ by:

$$(x^{(n)})_{n \in \mathbb{N} \setminus \{0\}} : \left\{ \begin{array}{l} x^{(1)} = x \\ x^{(n+1)} = x \diamond x^{(n)} \end{array} \right.$$

This is the sequence of the successive outputs of the node in the loop graph (Figure 4).

Definition 1 *The magma* $(\mathbb{S}, \diamond)$ *is weak sequence idempotent if it satisfies:*

$$\forall x \in \mathbb{S}, \quad \exists k_x \in \mathbb{N} \setminus \{0\} \qquad x^{(k_x+1)} = x^{(k_x)}$$

In the following, k_x denotes the smallest integer satisfying $x^{(k+1)} = x^{(k)}$. The following proposition is immediate:

Proposition 1 *The global computation on the loop graph (Figure 4) is defined on* $(\mathbb{S}, \diamond)$ *if and only if* $\diamond$ *is sequence idempotent*[1]*.*

For instance, the operator $\oslash_1$ defined on $\mathbb{N}$ by $x \oslash_1 y = \min(p, x + y)$ with $p \in \mathbb{N} \setminus \{0, 1\}$ is weak sequence idempotent, with $k_x = \lfloor \frac{p}{x} \rfloor + 1$. It then ensures the existence of the scour primitive on the loop graph.

In many cases, one can consider a unique constant k for all $x \in \mathbb{S}$:

Definition 2 *The magma* $(\mathbb{S}, \diamond)$ *is* k *sequence idempotent if it satisfies:*

$$\forall x \in \mathbb{S}, \quad x^{(k+1)} = x^{(k)}$$

For instance, the operator $\oslash_2$ defined on $\mathbb{N}$ by $x \oslash_2 y = \min(2x, x + y)$ is 2 sequence idempotent, and the operator $\oslash_1$ defined above by $x \oslash_1 y = \min(p, s + y)$ is p sequence idempotent on $\mathbb{N} \setminus \{0\}$. An idempotent operator is 1 sequence idempotent: $x^{(2)} = x \diamond x = x = x^{(1)}$.

Proposition 2 *A* k *sequence idempotent operator is weak sequence idempotent.*

Proof. $\forall x \in \mathbb{S}$, $k_x = k$ is convenient. □

Lemma 1 *If* $x^{(k+1)} = x^{(k)}$*, then for any* $p \in \mathbb{N}$*,* $x^{(k+p)} = x^{(k)}$*.*

Proof. Suppose that this is true for l: $x^{(k+l)} = x^{(k)} = x^{(k+1)}$. Then we have $x^{(k+l+1)} = x \diamond x^{(k+l)} = x \diamond x^{(k)} = x^{(k+1)} = x^{(k)}$ and this is true for $l + 1$. Since this is true for $p = 0$ and $p = 1$, then by recurrence $x^{(k+p)} = x^{(k)}$, for any $p \in \mathbb{N}$. □

---

[1] In this report, we indifferently write "$\diamond$ is weak sequence idempotent on $\mathbb{S}$" or "the magma $(\mathbb{S}, \diamond)$ is weak sequence idempotent", and this is the same for others properties.

Proposition 3 *A k sequence idempotent operator is $k+p$ sequence idempotent for any $p \in \mathbb{N}$.*

Proof. By Lemma 1. $\square$

Proposition 4 *If the magma $(\mathbb{S}, \diamond)$ is finite and weak sequence idempotent, then there exists an integer $k \in \mathbb{N}$ such that it is k sequence idempotent.*

Proof. Let k be the integer defined by $k = \max\{k_x, x \in \mathbb{S}\}$. Then, for any $x \in \mathbb{S}$, since $x^{(k_x)} = x^{(k_x+1)}$ and $k - k_x \geq 0$, we have by Lemma 1 $x^{(k)} = x^{(k_x+(k-k_x))} = x^{(k_x)}$ and similarly $x^{(k+1)} = x^{(k_x)}$. Thus, for any $x \in \mathbb{S}$, $x^{(k+1)} = x^{(k)}$ and $\diamond$ is k sequence idempotent. $\square$

3.2 Order idempotency

We introduce other forms of idempotency, that can be considered instead of the sequence idempotency when the brackets can be suppressed in the expression $x \diamond (x \diamond (\cdots (x \diamond x) \cdots)$. Let define the sequence $(x^n)_{n \in \mathbb{N} \setminus \{0\}}$ by $x^1 = x$ and $x^{n+1} = x^n \diamond x$.

Definition 3 *The magma $(\mathbb{S}, \diamond)$ is weak order idempotent if it satisfies:*

$$\forall x \in \mathbb{S}, \quad \exists k'_x \in \mathbb{N} \setminus \{0\} \quad x^{k'_x+1} = x^{k'_x}$$

In the following, k'_x denotes the smallest integer satisfying $x^{k'_x+1} = x_{k'_x}$.

Proposition 5 *Let $(\mathbb{S}, \diamond)$ be an associative magma (monoid). Then it is weak sequence idempotent if and only if it is weak order idempotent.*

Proof. This is because $x^{(n)} = x^n$ by associativity. $\square$

Proposition 6 *Let $(\mathbb{S}, \diamond)$ be a commutative magma. Then it is weak sequence idempotent if and only if it is weak order idempotent.*

Proof. This is because $x^{(n)} = x^n$ by commutativity. Indeed, this is true for $n = 1$ and $n = 2$. Suppose that this is true for n. Then, by commutativity, we have: $x^{(n+1)} = x \diamond x^{(n)} = x^{(n)} \diamond x = x^n \diamond x = x^{n+1}$. Then, by recurrence, $x^{(n)} = x^n$ for any $x \in \mathbb{S}$ and non nul integer n. $\square$

As with the sequence idempotency, in many cases a single constant k can be considered.

Definition 4 *The magma $(\mathbb{S}, \diamond)$ is k* order idempotent *with $k \in \mathbb{N} \setminus \{0\}$ if it satisfies:*

$$\forall x \in \mathbb{S}, \quad x^{k+1} = x^k$$

As for the sequence idempotency, we have the following three propositions (with similar proofs):

Proposition 7 *If the operator $\diamond$ is k order idempotent, then it is weak order idempotent.*

Proposition 8 *If the operator $\diamond$ is k order idempotent, then it is $k+p$ order idempotent for any $p \in \mathbb{N}$.*

Proposition 9 *If the magma $(\mathbb{S}, \diamond)$ is finite and weak order idempotent, then there exists an integer $k \in \mathbb{N}$ such that it is k order idempotent.*

An idempotent operator is 1 order idempotent, and then weak order idempotent, and also k order idempotent for any $k \in \mathbb{N} \setminus \{0\}$. But a k order idempotent operator is not necessarily idempotent. For instance, the operator $\oslash_1$ (defined on $\mathbb{N}$ by $x \oslash_1 y = \min(p, x+y)$ with $p \in \mathbb{N} \setminus \{0, 1\}$), is p order idempotent, but not idempotent ($1 \oslash_1 1 \neq 1$).

Finally, we have the following relations (same proofs as Propositions 5 and 6):

Proposition 10 *Let $(\mathbb{S}, \diamond)$ be an associative magma (monoid). Then it is k sequence idempotent if and only if it is k order idempotent.*

Proposition 11 *Let $(\mathbb{S}, \diamond)$ be a commutative magma. Then it is k sequence idempotent if and only if it is k order idempotent.*

For instance, the operator $\oslash_2$ (defined on $\mathbb{N}$ by $x \oslash_2 y = \min(2x, x+y)$) is not associative nor commutative. While it is 2 sequence idempotent, it is not k order idempotent, for any integer k.

3.3 Summary

In order the global computation terminates on the loop graph, the weak sequence idempotency is required. In a finite set, this is equivalent to the k sequence idempotency. In the special case of an associative or commutative magma, the (weak or k) sequence idempotency is equivalent to the (weak or k) order idempotency. This is because associativity or commutativity allow to suppress brackets, leading to $x^{(k)} = x^k$. However, we will see that other properties allow to suppress brackets.

4 Weak left cancellation

Since some forms of idempotency is required to define the scour primitive, no inverse element can be used to simplify the equations. However, we introduce a new property, which is generally satisfied by the operators, and which allows some simplifications in the expressions.

Let y, z be two elements of $\mathbb{S}$ endowed by $\diamond$. We have:

$$(y = z) \quad \Rightarrow \quad (\forall x \in \mathbb{S}, x \diamond y = x \diamond z) \tag{2}$$

but the reciprocal could be false. To be convinced of that, let consider the silly operator $\oslash_3$ defined by $x \oslash_3 y = x$ for any $x, y \in \mathbb{S}$.

Definition 5 *A magma* $(\mathbb{S}, \diamond)$ *is* weak left cancellative *(wlc for short) if it satisfies:*

$$\forall y, z \in \mathbb{S}, \qquad (\forall x \in \mathbb{S}, x \diamond y = x \diamond z) \quad \Leftrightarrow \quad (y = z)$$

Most usual operators are weak left cancellative on their definition set. This is in fact a really raisonnable hypothesis that could be interpreted as follows: "if no element of $\mathbb{S}$ disagrees with the fact that y could be equal to z, then it is established that $y = z$".

To point out the difference between the left cancellative and the weak left cancellative property, let consider the set of integers $\mathbb{N}$ endowed by the min operator. If $(\mathbb{N}, \min)$ is left cancellative, then:

$$\forall x, y, z \in \mathbb{N}, \qquad (\min(x, y) = \min(x, z)) \quad \Rightarrow \quad (y = z)$$

But this is false with $x = 2$, $y = 3$ and $z = 4$. Now, if $(\mathbb{N}, \min)$ is weak left cancellative, then:

$$\forall y, z \in \mathbb{N}, \qquad (\forall x \in \mathbb{N}, \min(x, y) = \min(x, z)) \quad \Rightarrow \quad (y = z)$$

This is always true. Indeed, if $\min(x, y) = \min(x, z)$ is true for any $x \in \mathbb{N}$, by considering $x = y$ and $x = z$ we obtain $\min(y, y) = \min(y, z) = \min(z, z)$, and then $y = z$.

This example shows that an operator can be both wlc and idempotent, while it is not the case with the left cancellative property. Hence, while idempotent semigroups are not regular, some simplifications in equations will be possible, on the basis of the weak left cancellation property.

The following proposition states that any usual algebraic structure is weak left cancellative (such as semi-groups and groups):

Proposition 12 *If* $(\mathbb{S}, \diamond)$ *admits a left identity element, then it is weak left cancellative.*

Finally, Hypothesis 2 states that the operator $\diamond$ admits a right identity element. By weak left cancellative, it is unique:

Proposition 13 *If* $(\mathbb{S}, \diamond)$ *is weak left cancellative, then it admits at most one right identity elements.*

Proof. Suppose that $e^1_\diamond$ and $e^2_\diamond$ are two right identity elements of $(\mathbb{S}, \diamond)$. Then they verify: $\forall x \in \mathbb{S}, x \diamond e^1_\diamond = x = x \diamond e^2_\diamond$. By weak left cancellation, we have $e^1_\diamond = e^2_\diamond$. $\square$

5 Rank properties

As shown in Figure 1, the private data of a node appears always at the beginning of its local calculations, while other inputs appear on the right. This characteristic suggests some properties of the operators that could only apply after a certain rank in the expressions. In this section, we detail weaker properties of the classical commutativity, idempotency and associativity properties, based on the rank from which they apply in the expressions.

5.1 Rank commutativity

5.1.1 Rank k commutativity

To fulfil the Requirement 1, meaning that the global computation should give the same result in any representation of a given graph (see for instance Figure 3 page 5), the operators should not be aware of the order of the inputs on each node. We then introduce the following property on binary operators.

Definition 6 *Let k be a non nul integer, and $(\mathbb{S}, \diamond)$ be a magma. Then $\diamond$ is rank k commutative if it satifies:*

$$\forall x_1, \ldots, x_{k+1} \in \mathbb{S}, \qquad x_1 \diamond \cdots \diamond x_{k-1} \diamond x_k \diamond x_{k+1} \quad = \quad x_1 \diamond \cdots \diamond x_{k-1} \diamond x_{k+1} \diamond x_k$$

A commutative operator is rank 1 commutative. The following proposition is immediate, by considering a leaves-to-root oriented tree of three nodes.

Proposition 14 *Let $(\mathbb{S}, \diamond)$ be a magma which is not rank 2 commutative. Then a global computation relying on this magma does not satisfy the Requirement 1.*

5.1.2 Rank 2 commutativity

The following propositions concern some rank related relations.

Proposition 15 *Let $(\mathbb{S}, \diamond)$ be a magma that is rank k commutative with $k \geq 2$. If it admits a right identity element $e_\diamond$, then it is rank 2 commutative.*

Proof. We have:

$$x_1 \diamond x_{k+1} \diamond x_k = x_1 \diamond \underbrace{e_\diamond \diamond \cdots \diamond e_\diamond}_{k-2} \diamond x_{k+1} \diamond x_k = x_1 \diamond \underbrace{e_\diamond \diamond \cdots \diamond e_\diamond}_{k-2} \diamond x_k \diamond x_{k+1} = x_1 \diamond x_k \diamond x_{k+1}.$$

□

Proposition 16 *If the operator $\diamond$ is rank k commutative with $k \geq 2$, then it is rank $k+p$ commutative for any $p \in \mathbb{N}$.*

Proof. $x_1 \diamond \cdots \diamond x_{k+p+1} = (x_1 \diamond \cdots \diamond x_{p+1}) \diamond x_{p+2} \diamond \cdots \diamond x_{p+k-1} \diamond x_{p+k} \diamond x_{p+k+1} = (x_1 \diamond \cdots \diamond x_{p+1}) \diamond x_{p+2} \diamond \cdots \diamond x_{p+k-1} \diamond x_{p+k+1} \diamond x_{p+k} = x_1 \diamond \cdots \diamond x_{k+p-1} \diamond x_{k+p+1} \diamond x_{k+p}$. $\square$

For instance, let define the $\oslash_4$ operator on the integers by $x \oslash_4 y = x+2y$. It is not commutative but it is rank 2 commutative: $x \oslash_4 y \oslash_4 z = x + 2y + 2z = x+2z+2y = x \oslash_4 z \oslash_4 y$. It is also rank k commutative for any integer $k \geq 2$ (Proposition 16).

The previous proposition is not true for $k = 1$: if an operator is commutative, then it is rank k commutative for $k = 1$, but not for $k > 1$, as we will see below.

Thanks to the two previous propositions, and since we admit Hypothesis 2 (which means that there exists a right identity element), we will only consider rank 2 commutativity in the following.

5.1.3 Usefulness of rank 2 commutativity

We now present the relations between associativity and rank commutativity.

Proposition 17 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative monoid (i.e., associative magma). Then, if it is rank 2 commutative, it is commutative.*

Proof. For any $x, y, z \in \mathbb{S}$, we have by rank 2 commutativity $x \diamond y \diamond z = x \diamond z \diamond y$. By associativity, we then have $x \diamond (y \diamond z) = x \diamond (z \diamond y)$. By weak left cancellation, we then have: $y \diamond z = z \diamond y$, and the monoid is commutative. $\square$

Proposition 18 *An associative and commutative operator is rank k commutative for any $k \geq 1$.*

Proof. By definition, a commutative operator is rank 1 commutative. For any $x, y, z \in \mathbb{S}$, $x \diamond y \diamond z = x \diamond (y \diamond z)$ by associativity. Then $x \diamond y \diamond z = x \diamond (z \diamond y)$ by commutativity. Thus $x \diamond y \diamond z = x \diamond z \diamond y$ by associativity and the operator is then rank 2 commutativity. By Proposition 16, it is then rank k commutative for any $k \geq 1$. $\square$

Note that associativity is necessary in the previous proof. Indeed, let consider the $\oslash_5$ operator defined on the set $\{x, y, z\}$ by:

$\oslash_5$	x	y	z
x	x	z	y
y	z	y	x
z	y	x	z

The operator $\oslash_5$ is commutative but not associative. It is not rank 2 commutative: $x \oslash_5 y \oslash_5 z = z$ and $x \oslash_5 z \oslash_5 y = y$.

Hence, as stated by Propositions 17 and 18, the rank 2 commutativity property is useful only if the operator is not associative. We will see that interesting non associative operators can be used for the global computations.

5.1.4 Summary

If the operator is associative and commutative, then the Requirement 1 is satisfied. But we will see that weaker hypotheses are sufficient for that. We showed however that the rank 2 commutativity is necessary to satisfy the Requirement 1.

5.2 Rank idempotency

5.2.1 Rank k idempotency

It is common to assimilate the computation of a node as the building of an expression, composed with the private data of the node, its inputs (considered as expressions) and its local operator. To any finite expression composed of values of $\mathbb{S}$, binary operators defined on $\mathbb{S}$ and brackets, corresponds a specific acyclic directed graph with a single sink, such that the given expression is the output of the sink. When the graph admits some circuit, some infinite expressions could be obtained In order the Requirement 2 is satisfied, any output should correspond to a *finite* expression even in presence of circuits. In other words, the operators should have the ability to *reduce the expressions*, such that any infinite expression can be reduced to some finite one.

We introduce the rank k idempotency property, which states that, from a certain rank in the expressions, some reductions can be done.

Definition 7 *Let $\mathbb{S}$ be a set endowed by the law $\diamond$. The operator $\diamond$ is* rank k idempotent *(with $k \in \mathbb{N} \setminus \{0\}$) if it satisfies:*

$$\forall x_1, \ldots, x_k, \qquad x_1 \diamond \cdots \diamond x_k \diamond x_k \quad = \quad x_1 \diamond \cdots \diamond x_k$$

An idempotent operator is rank 1 idempotent, but not necessarily rank 2 idempotent. For instance, the $\oslash_6$ operator defined on the integers by $x \oslash_6 y = x$ if $x < y$ and $x \oslash_6 y = 0$ if $x \geq y$ is not idempotent but it is rank 2 idempotent:

$$x \oslash_6 y = \begin{cases} 0 & \text{if } x \geq y \\ x & \text{if } x < y \end{cases}$$

Then we have:

$$\begin{aligned}
x \oslash_6 y \oslash_6 y &= \begin{cases} 0 \oslash_6 y & \text{if } x \geq y \\ x \oslash_6 y & \text{if } x < y \end{cases} \\
&= \begin{cases} 0 & \text{if } x \geq y & \text{and } 0 \geq y \\ 0 & \text{if } x \geq y & \text{and } 0 < y \\ 0 & \text{if } x < y & \text{and } x \geq y \\ x & \text{if } x < y & \text{and } x < y \end{cases} \\
&= \begin{cases} 0 & \text{if } x \geq y \\ x & \text{if } x < y \end{cases} = x \oslash_6 y \\
&= x \oslash_6 y
\end{aligned}$$

5.2.2 Rank 2 idempotency

The following propositions concern some rank related properties.

Proposition 19 *Let* $(\mathbb{S}, \diamond)$ *be a magma. If* $\diamond$ *is rank* k *idempotent with* $k > 1$*, it is rank* $k + p$ *idempotent for any integer* $p \in \mathbb{N}$.

Proof. This is because $x_1 \diamond \cdots \diamond x_{k+p} \diamond x_{k+p} = (x_1 \diamond \cdots \diamond x_{p+1}) \diamond \cdots \diamond x_{k+p} \diamond x_{k+p}$. □

Proposition 20 *Let* $(\mathbb{S}, \diamond)$ *be a magma admitting a right identity element* $e_\diamond$*. Then, if it is rank* k *idempotent, it is rank 2 idempotent.*

Proof. For any $x, y \in \mathbb{S}$, $x \diamond y \diamond y = x \diamond \underbrace{e_\diamond \diamond \cdots \diamond e_\diamond}_{k-2} \diamond y \diamond y = x \diamond \underbrace{e_\diamond \diamond \cdots \diamond e_\diamond}_{k-2} \diamond y = x \diamond y$. □

Thanks to these two previous propositions, and since we admitted Hypothesis 2, we will only consider the rank 2 idempotency in the following.

5.2.3 Usefulness of rank 2 idempotency

We now present the relations between associativity and rank idempotency.

Proposition 21 *Let* $(\mathbb{S}, \diamond)$ *be a weak left cancellative monoid (i.e., associative magma). Then, if it is rank 2 idempotent, it is idempotent.*

Proof. For any $x, y \in \mathbb{S}$, we have by rank 2 idempotency $x \diamond y \diamond y = x \diamond y$. By associativity, we then have $x \diamond (y \diamond y) = x \diamond y$. By weak left cancellation, we then have: $y \diamond y = y$, and the monoid is idempotent. □

Proposition 22 *An associative and idempotent operator is rank* k *idempotent for any* $k \geq 1$.

Proof. By definition, an idempotent operator is rank 1 idempotent. For any $x, y \in \mathbb{S}$, $x \diamond y \diamond y = x \diamond (y \diamond y)$ by associativity. Then $x \diamond y \diamond y = x \diamond y$ by idempotency. Then $\diamond$ is rank 2 idempotent. By Proposition 19, it is then rank k idempotent for any $k \geq 1$. □

Hence, as with rank commutativity, the rank idempotency is only useful if the operator is not associative. The following proposition is obvious. It makes a link between the rank idempotent and the order idempotency.

Proposition 23 *Let* $(\mathbb{S}, \diamond)$ *be a magma. If* $\diamond$ *is rank* k *idempotent, it is* k *order idempotent.*

Hence, a rank 2 idempotent operator is also 2 order idempotent. And if brackets can be suppressed in expressions by means of associativity or commutativity (as in Propositions 10 or 11) or by other properties, then a rank 2 idempotent operator is also 2 sequence idempotent.

5.2.4 Summary

Thanks to the existence of a right identity element, we only consider rank 2 idempotency. This property is only useful in a non associative magma. A rank 2 idempotent operator is also 2 order idempotent. If some additional properties allow to suppress brackets, then the rank 2 idempotency will ensure the termination of the global computation on the loop graph (Figure 4).

5.3 Rank associativity

Both rank 2 commutativity and rank 2 idempotency properties have useful applications for the termination of the global computation. One could ask about usefulness of the rank associativity.

Definition 8 *A magma $(\mathbb{S},\diamond)$ is rank k associative if it satisfies:*

$$\forall x_1,\dots,x_{k+2}, \qquad x_1\diamond\cdots\diamond x_{k+2} \quad = \quad x_1\diamond\cdots\diamond x_k\diamond(x_{k+1}\diamond x_{k+2})$$

Proposition 24 *An associative operator is rank k associative for any $k \in \mathbb{N}^*$.*

Proof. This is because $x_1\diamond\cdots\diamond x_k\diamond(x_{k+1}\diamond x_{k+2}) = (x_1\diamond\cdots\diamond x_k)\diamond(x_{k+1}\diamond x_{k+2})$. By associativity, we then obtain $x_1\diamond\cdots\diamond x_k\diamond(x_{k+1}\diamond x_{k+2}) = x_1\diamond\cdots\diamond x_k\diamond x_{k+1}\diamond x_{k+2}$. $\square$

Definition 9 *A magma $(\mathbb{S},\diamond)$ is said two decomposable if, for any $x \in \mathbb{S}$, there exists $y, z \in \mathbb{S}$ such that $x = y\diamond z$.*

A magma with a right (resp. left) identity element $e_\diamond$ is two decomposable because for any $x \in \mathbb{S}$, we have $x = x\diamond e_\diamond$ (resp. $x = e_\diamond\diamond x$). An idempotent magma is two decomposable because for any $x \in \mathbb{S}$, we have $x = x\diamond x$.

Proposition 25 *Let $(\mathbb{S},\diamond)$ be a two decomposable magma. Then, if it is rank k associative, it is associative.*

Proof. For any $x \in \mathbb{S}$, by two decomposability, there exists $x_1, x'_1, x_2, x'_2, \dots, x_k, x'_k$ in $\mathbb{S}$ such that $x = x'_k\diamond x_k$, $x'_k = x'_{k-1}\diamond x_{k-1}$, $\dots$, $x'_2 = x'_1\diamond x_1$. Hence, $x\diamond(y\diamond z) = x'_k\diamond x_k\diamond(y\diamond z) = x'_{k-1}\diamond x_{k-1}\diamond x_k\diamond(y\diamond z) = x'_{k-2}\diamond x_{k-1}\diamond x_k$, and so one. We then obtain $x\diamond(y\diamond z) = x'_1\diamond x_1\diamond x_2\diamond\cdots\diamond x_k\diamond(y\diamond z)$. By rank k associativity, we then have $x\diamond(y\diamond z) = x'_1\diamond x_1\diamond x_2\diamond\cdots\diamond x_k\diamond y\diamond z$. Finally we obtain $x\diamond(y\diamond z) = x\diamond y\diamond z$ and $\diamond$ is associative on $\mathbb{S}$. $\square$

As a corollary, an idempotent or unifere rank k associative magma is associative, and since we admit Hypothesis 2, we do not consider rank associativity in the following.

5.4 Summary

Two rank properties are important for computing on networks: the rank 2 commutativity, and the rank 2 idempotency. The first one is necessary to obtain a result that does not depend on the drawing of the network (see Figure 3). The second one implies the rank k idempotency ($k \geq 2$) and then the order k idempotency, which is sufficient in certain conditions to ensure the termination of the global computation on the loop graph (see Figure 4). However, the rank k associativity is not useful because we admitted the existence of a right identity element. Nevertheless, other properties are needed to deal with brackets, as the commutativity or associativity do. We will now introduce weaker properties than commutativity and associativity.

6 r-properties

In this section, we introduce some new properties, generalizing the classical properties of associativity, commutativity and idempotency, by using an application. They are simpler to use than the rank properties, and allow to define the r-operators. We show the relations between these properties and the above discussed rank properties.

6.1 r-associativity

The r-associativity is a generalization of the associativity which is very powerful; we will often use it in further proofs.

Definition 10 *Let* $(\mathbb{S}, \diamond)$ *be a magma, and* $r : \mathbb{S} \to \mathbb{S}$ *an application. The operator* $\diamond$ *is* r-associative *if it satisfies:*

$$\forall x, y, z \in \mathbb{S}, \qquad x \diamond (y \diamond z) = x \diamond y \diamond r(z)$$

For instance, the operator $\oslash_4$ (defined on $\mathbb{N}$ by $x \oslash_4 y = x + 2y$) is r-associative, with the mapping $r : x \mapsto 2x$. An associative operator is r-associative with the identity mapping $r : x \mapsto x$.

The following propositions concern the properties of the application r from $\mathbb{S}$ to $\mathbb{S}$, providing that $(\mathbb{S}, \diamond)$ is a r-associative magma.

Proposition 26 *Let* $(\mathbb{S}, \diamond)$ *be a weak left cancellative magma. If there exists an application* $r : \mathbb{S} \to \mathbb{S}$ *such that* $\diamond$ *is r-associative, then* r *is injective.*

Proof. For any $a, b, x, y \in \mathbb{S}$, if $r(x) = r(y)$, we have $a \diamond b \diamond r(x) = a \diamond b \diamond r(y)$. Then, by r-associativity, we have $a \diamond (b \diamond x) = a \diamond (b \diamond y)$. Then, by weak left cancellation, we have $b \diamond x = b \diamond y$ and finally $x = y$. $\square$

Note that the weak left cancellation is necessary. Indeed, let consider the operator $\oslash_7$ defined on $\mathbb{N}$ by $x \oslash_7 y \equiv \min(x, \min(k, y+1))$, with $k \in \mathbb{N}$. It is r-associative with the application $r : x \mapsto \min(k, x+1)$. However $(\mathbb{N}, \oslash_7)$ is not weak left cancellative, and the application r is not injective.

Proposition 27 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma admitting the right identity element $e_\diamond$. If there exists an application $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative, then r is defined by $r(x) = e_\diamond \diamond x$ for any $x \in \mathbb{S}$.*

Proof. For any $x, y \in \mathbb{S}$, $x \diamond r(y) = x \diamond e_\diamond \diamond r(y)$. Then, by r-associativity, we have $x \diamond r(y) = x \diamond (e \diamond y)$. By weak left cancellation, we obtain $r(y) = e \diamond y$. □

As a consequence, we have the following corollaries:

Corollary 28 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma admitting the right identity element $e_\diamond$. If there exists an application $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative, then r is unique.*

Proof. By Proposition 13, $e_\diamond$ is unique. Then, by Proposition 27, r is unique. □

Corollary 29 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma admitting the right identity element $e_\diamond$. If there exists an application $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative, then $r(e_\diamond) = e_\diamond$, and for any $x \in \mathbb{S} \setminus \{e_\diamond\}$, $r(x) \neq e_\diamond$.*

Proof. By Proposition 27, $r(e_\diamond) = e_\diamond \diamond e_\diamond = e_\diamond$. Suppose that there exists $x \in \mathbb{S} \setminus \{e_\diamond\}$ such that $r(x) = e_\diamond$. Then $r(x) = r(e_\diamond)$ and r is not injective. This contradicts Proposition 26. □

The following proposition is immediate.

Proposition 30 *Let $(\mathbb{S}, \diamond)$ be a magma. If $r : \mathbb{S} \to \mathbb{S}$ is an homomorphism, then $(r(\mathbb{S}), \diamond)$ is a magma.*

Proof. This is because for any $x, y \in \mathbb{S}$, we have $r(x) \diamond r(y) = r(x \diamond y) \in r(\mathbb{S})$. □

Its reciprocal is false. Indeed, consider the magma $(\mathbb{N}, +)$ and the application r on the integers defined by: $r(x) = x$ if x is even, and $r(x) = x - 1$ is x is odd. While $(r(\mathbb{N}), +)$ is a magma (even integers endowed by the addition), r is not an homomorphism because $r(3+3) = r(6) = 6$ and $r(3) + r(3) = 2 + 2 = 4$. However the following proposition holds.

Proposition 31 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma, and $r : \mathbb{S} \to \mathbb{S}$ an application such that $\diamond$ is r-associative. If $(r(\mathbb{S}), \diamond)$ is a magma, then r is an endomorphism of $(\mathbb{S}, \diamond)$.*

Proof. For any x, y, u, v in $\mathbb{S}$, by r-associativity we have: $x \diamond y \diamond r(u) \diamond r^2(v) = (x \diamond y) \diamond r(u) \diamond r^2(v) = (x \diamond y) \diamond (r(u) \diamond r(v)) = x \diamond y \diamond (r(u) \diamond r(v))$. Since $r(\mathbb{S})$ is a magma, there exists $t \in \mathbb{S}$ such that $r(u) \diamond r(v) = r(t)$. Hence, $x \diamond y \diamond (r(u) \diamond r(v)) = x \diamond y \diamond r(t) = x \diamond (y \diamond t)$.

On an other hand, still using r-associativity, we have: $x \diamond y \diamond r(u) \diamond r^2(v) = x \diamond (y \diamond u) \diamond r^2(v) = x \diamond ((y \diamond u) \diamond r(v)) = x \diamond (y \diamond u \diamond r(v)) = x \diamond (y \diamond (u \diamond v))$.

We then have $x \diamond (y \diamond t) = x \diamond (y \diamond (u \diamond v))$. By weak left cancellation, we obtain first $y \diamond t = y \diamond (u \diamond v)$, and next $t = u \diamond v$. Thus $r(t) = r(u \diamond v) = r(u) \diamond r(v)$ and r is an endomorphism of $(\mathbb{S}, \diamond)$. $\square$

The following proposition is immediate; its reciprocal is false (consider the parity function on $(\mathbb{N}, \times)$, that returns 0 for even integers and 1 for odd integers).

Proposition 32 *Let $(\mathbb{S}, \diamond)$ be a magma. If $r : \mathbb{S} \to \mathbb{S}$ is a surjection, then $(r(\mathbb{S}), \diamond)$ is a magma.*

Proof. Since r is a surjection, for any $x, y \in \mathbb{S}$, there exists $z \in \mathbb{S}$ such that $r(x) \diamond r(y) = r(z)$. This means that $(r(\mathbb{S}), \diamond)$ is a magma. $\square$

We then have the following corollary:

Corollary 33 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma, and $r : \mathbb{S} \to \mathbb{S}$ a surjection such that $\diamond$ is r-associative. Then r is an automorphism of $(\mathbb{S}, \diamond)$ (that is a bijective homomorphism from $(\mathbb{S}, \diamond)$ to itself).*

Proof. By Proposition 32, since r is a surjection, $(r(\mathbb{S}), \diamond)$ is a magma. By Proposition 31, r is then an endomorphism of $(\mathbb{S}, \diamond)$. By Proposition 26, since $\diamond$ is r-associative, r is injective. Thus r is an injective and surjective endomorphism, that is an automorphism of $(\mathbb{S}, \diamond)$. $\square$

Note that r could be an endomorphism without being surjective. For instance, the operator $\oslash_8$ defined on $\mathbb{N}$ by $x \oslash_8 y = \min(x, 2y)$ is r-associative with $r : x \mapsto 2x$. This application is injective but not surjective ($r(\mathbb{N})$ is the set of even integers). Since $r(x) \oslash_8 r(y) = \min(2x, 4y)$ is even, $(r(\mathbb{N}), \oslash_8)$ is a magma. We have $r(x \oslash_8 y) = \min(2x, 4y) = r(x) \oslash_8 r(y)$ and then r is an endomorphism of $(\mathbb{N}, \oslash_8)$.

The r-associativity is a new important basic property, in the sense that none of the other properties implies it. As we will see, it has many applications.

6.2 r-commutativity

As for r-associativity, we introduce a generalization of the classical property of commutativity using an application.

Definition 11 *Let $(\mathbb{S}, \diamond)$ be a magma, and $r : \mathbb{S} \to \mathbb{S}$ an application. The operator $\diamond$ is* r-commutative *if it satisfies:*

$$\forall x, y \in \mathbb{S}, \qquad r(x) \diamond y = r(y) \diamond x$$

For instance, the operator $\oslash_4$ defined on $\mathbb{N}$ by $x \oslash_4 y = x + 2y$ is not commutative. But it is r-commutative with the mapping $r : x \mapsto 2x$. A commutative operator is r-commutative with the identity mapping: $r : x \mapsto x$.

We now present the connections between r-commutativity and rank commutativity, using the r-associativity.

Proposition 34 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma, and $r : \mathbb{S} \to \mathbb{S}$ an application. If $\diamond$ is r-associative and rank 2 commutative on $r(\mathbb{S})$, then it is r-commutative.*

Proof. For any $x, y, z \in \mathbb{S}$, we have $x \diamond r(y) \diamond r(z) = x \diamond r(z) \diamond r(y)$ by rank 2 commutativity. By r-associativity, we have $x \diamond r(y) \diamond r(z) = x \diamond (r(y) \diamond z)$ and $x \diamond r(z) \diamond r(y) = x \diamond (r(z) \diamond y)$. Thus, by weak left cancellation, we obtain $r(y) \diamond z = r(z) \diamond y$ for any $y, z \in \mathbb{S}$. $\square$

Proposition 35 *Let $(\mathbb{S}, \diamond)$ be a magma. If $\diamond$ is r-associative and r-commutative, then it is rank 2 commutative on $r(\mathbb{S})$.*

Proof. Let $u, v, w \in \mathbb{S}$ such that $r(u) = x$, $r(v) = y$ and $r(w) = z$. By r-associativity, we have $x \diamond y \diamond z = x \diamond (y \diamond w)$. By r-commutativity, we have $x \diamond (y \diamond w) = x \diamond (r(u) \diamond w) = x \diamond (r(w) \diamond u)$. By r-associativity, we have $x \diamond (r(w) \diamond u) = x \diamond r(w) \diamond r(u) = x \diamond z \diamond y$. $\square$

Corollary 36 *Let $(\mathbb{S}, \diamond)$ be a magma. If there exists a surjective mapping $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative and r-commutative with this application, then $\diamond$ is rank 2 commutative.*

Hence, this corollary makes a link between some r-properties and the rank 2 commutativity. However the same result is obtained with an homorphism in a weak left cancellative magma:

Proposition 37 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma. If there exists an homomorphism $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative and r-commutative, then it is rank 2 commutative.*

Proof. Let $x, y, z \in \mathbb{S}$. By homomorphism and then r-associativity, we have $r(x \diamond y \diamond z) = r(x) \diamond r(y) \diamond r(z) = r(x) \diamond (r(y) \diamond z)$. By r-commutativity and then r-associativity, we have $r(x) \diamond (r(y) \diamond z) = r(x) \diamond (r(z) \diamond y)$. By r-associativity and homomorphism, we have: $r(x)\diamond(r(z)\diamond y) = r(x)\diamond r(z)\diamond r(y) = r(x\diamond z\diamond y)$. Hence, for any $x, y, z \in \mathbb{S}$, $r(x\diamond y\diamond z) = r(x\diamond z\diamond y)$.

By Proposition 26, r is injective because $(\mathbb{S}, \diamond)$ is wlc and $\diamond$ is r-associative. Thus $r(x\diamond y\diamond z) = r(x\diamond z\diamond y)$ implies that $x\diamond y\diamond z = x\diamond z\diamond y$, and $\diamond$ is rank 2 commutative. □

6.3 r-idempotency

As for r-associativity and r-commutativity, we introduce a generalization of the classical property of idempotency using an application.

Definition 12 *Let $(\mathbb{S}, \diamond)$ be a magma, and $r : \mathbb{S} \to \mathbb{S}$ an application. The operator $\diamond$ is* r-idempotent *if it satisfies:*

$$\forall x \in \mathbb{S}, \qquad r(x)\diamond x = r(x)$$

We now present the connections between this new property and the rank idempotency.

Proposition 38 *Let $(\mathbb{S}, \diamond)$ be a magma. If there exists an application $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative and r-idempotent, then for any $x \in \mathbb{S}$ and any $y \in r(\mathbb{S})$, we have: $x\diamond y\diamond y = x\diamond y$.*

Proof. Let $x, y, v \in \mathbb{S}$ such that $r(v) = y$. By r-associativity, we have: $x\diamond y\diamond y = x\diamond r(v)\diamond r(v) = x\diamond(r(v)\diamond v)$. By r-idempotency, we have: $x\diamond(r(v)\diamond v) = x\diamond r(v)$. Thus $x\diamond y\diamond y = x\diamond y$. □

We call this last property rank 2 idempotency on $r(\mathbb{S})$.

Proposition 39 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma, and $r : \mathbb{S} \to \mathbb{S}$ an application. If $\diamond$ is r-associative and rank 2 idempotent on $r(\mathbb{S})$, then it is r-idempotent.*

Proof. For any $x, y \in \mathbb{S}$, we have $x\lhd r(y)\lhd r(y) = x\lhd(r(y)\lhd y)$ by r-associativity. By rank 2 idempotency on $r(\mathbb{S})$, we have: $x\lhd r(y)\lhd r(y) = x\lhd r(y)$. Then we have $x\lhd(r(y)\lhd y) = x\lhd y$ and by weak left cancellation, the following holds: $r(y)\lhd y = y$. Hence $\lhd$ is r-idempotent on $\mathbb{S}$. □

We have the following corollaries:

Corollary 40 *Let $(\mathbb{S}, \diamond)$ be a magma. If there exists an application $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative and r-idempotent, then $\diamond$ is rank 2 idempotent on $r(\mathbb{S})$.*

Corollary 41 *Let $(\mathbb{S}, \diamond)$ be a magma. If there exists a surjection $r : \mathbb{S} \to \mathbb{S}$ such that $\diamond$ is r-associative and r-idempotent, then $\diamond$ is rank 2 idempotent.*

However, as we already seen, the application r could be an endomorphism of $(\mathbb{S}, \diamond)$ without being a surjection, and this is sufficient to ensure the rank 2 idempotency:

Proposition 42 *Let $(\mathbb{S}, \diamond)$ be a weak left cancellative magma. If there exists an endomorphism r of $(\mathbb{S}, \diamond)$ such that $\diamond$ is r-associative and r-idempotent, then $\diamond$ is rank 2 idempotent on $\mathbb{S}$.*

Proof. For any $x, y \in \mathbb{S}$, by homomorphism, we have: $r(x \diamond y \diamond y) = r(x) \diamond r(y) \diamond r(y)$. Then, by r-associativity, we have: $r(x \diamond y \diamond y) = r(x) \diamond (r(y) \diamond y)$. Then we have $r(x \diamond y \diamond y) = r(x) \diamond r(y)$ by r-idempotency, and finally $r(x \diamond y \diamond y) = r(x \diamond y)$ by homomorphism. Since $\diamond$ is weak left cancellative, r is injective (Proposition 26), and $x \diamond y \diamond y = x \diamond y$. □

7 r-operators

By combining the previous r-properties that generalize classical associativity, commutativity and idempotency, we now define a new algebraic structure, called r-semi-group. This structure encompasses the well known semi-groups, and has interesting properties for global computations.

7.1 r-semi-group

Definition 13 (r-semi-group) *Let $(\mathbb{S}, \triangleleft)$ be a weak left cancellative magma admitting the right identity $e_\triangleleft$ element, and let $r : \mathbb{S} \to \mathbb{S}$ be an endomorphism. Then $(\mathbb{S}, \triangleleft)$ is an* r-semi-group *if it is r-associative, r-commutative, r-idempotent with the application r.*

The law of an r-semi-group is called *r-operator*. The endomorphism r is called *r-mapping*. Figure 6 precises the place of the r-semi-group in the algebraic structures with one law. Note that the previously introduced operator $\oslash_7$ fulfils all the above conditions but the weak left cancellation and the right identity element; this is not an r-operator. The minc operator defined on $\mathbb{N} \cup \{+\infty\}$by $\text{minc}(x, y) = \min(x, y + 1)$ is an r-operator with the mapping $r : x \mapsto x + 1$.

Proposition 43 *Let $(\mathbb{S}, \triangleleft)$ be an r-semi-group. Then the right identity element $e_\triangleleft$ is unique.*

Proof. By Proposition 13, since $(\mathbb{S}, \triangleleft)$ is weak left cancellative. □

The following proposition is a consequence of the r-associativity, the weak left cancellation and the existence of $e_\triangleleft$.

Proposition 44 *Let $(\mathbb{S}, \triangleleft)$ be an r-semi-group. Then the following holds: (i) r is unique and satisfies $\forall x \in \mathbb{S}$, $r(x) = e_\triangleleft \triangleleft x$, (ii) r is injective, (iii) $\forall x \in \mathbb{S}$, $r(x) = e_\triangleleft \Leftrightarrow x = e_\triangleleft$.*

Proof. (i) is given by Proposition 27 and Corollary 28. (ii) is given by Proposition 26. (iii) is given by Proposition 29. □

The following proposition is given by the r-associativity, the r-commutativity, the homomorphism and the weak left cancellation.

Proposition 45 *Let $(\mathbb{S}, \triangleleft)$ be an r-semi-group. Then $\triangleleft$ is rank 2 commutative on $\mathbb{S}$.*

Proof. By Proposition 37. □

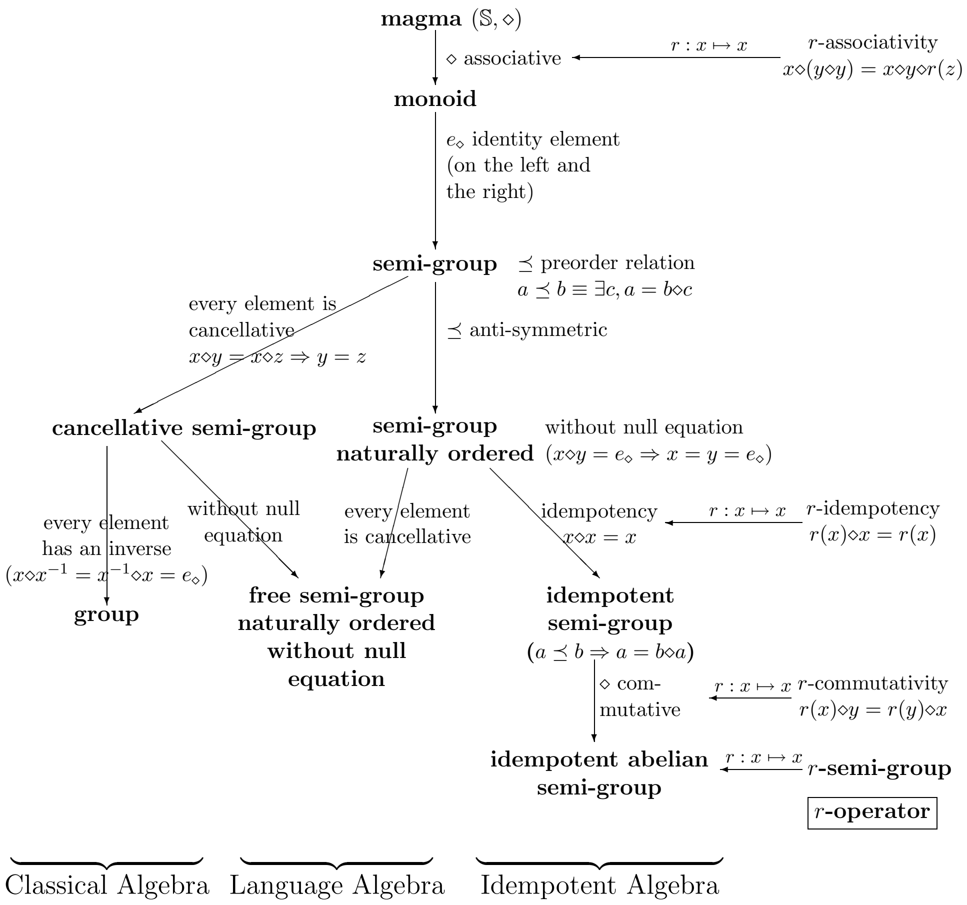

Figure 6: Place of the r-semi-group in the algebraic structures with one law.

The following proposition is given by the r-associativity, the r-idempotency, the homomorphism and the weak left cancellation.

Proposition 46 *Let* $(\mathbb{S}, \triangleleft)$ *be an r-semi-group. Then* $\triangleleft$ *is rank 2 idempotent on* $\mathbb{S}$*.*

Proof. By Proposition 42. □

This shows that, while defined with an homomorphism, an r-operator is still rank 2 commutative and rank 2 idempotent.

7.2 Surjective mapping versus endormorphism

In previous works [9], the r-operators relied on a surjective mapping (bijective by Proposition 26). However, while using an homomorphism, we obtain the same properties which are useful for computing on networks: rank 2 commutativity and rank 2 idempotency (Propositions 45 and 46).

For instance, the already introduced $\oslash_8$ operator (defined on $\mathbb{N}$ by $x \oslash_8 y = \min(x, 2y)$) is an r-operator on $\mathbb{N}$ with the mapping $x \mapsto 2x$, which is an endomorphism of $(\mathbb{N}, \oslash_8)$ but not a surjective mapping on $\mathbb{N}$ because odd integers are not reached.

Hence, the definition based on an endomorphism is more general. This is also an interesting improvement for computation on finite sets when the application is increasing (so-called strictly idempotent r-operators [9]). Note that by Corollary 33, if r is surjective then r is an endomorphism, but the converse is false.

For instance, let consider the already introduced minc r-operator (defined on $\mathbb{N} \cup \{+\infty\}$ by $\mathrm{minc}(x, y) = \min(x, y + 1)$ with $r : x \mapsto x + 1$). The mapping $r : x \mapsto x + 1$ is a surjective mapping on $\mathbb{N} \cup \{+\infty\}$, and then an endomorphism. But if we define minc on the empty set $\{0, 1, \ldots, 255\}$ with $r(x) = \min(255, x+1)$, the mapping is no more surjective (0 is not reached), but is still an endomorphism. Indeed, we have:

$$\begin{aligned} r(\mathrm{minc}(x, y)) &= r\left(\min(x, \min(255, y + 1))\right) \\ &= \min\left(255, \min(x, \min(255, y + 1)) + 1\right) \\ &= \min(255, x + 1, y + 2) \end{aligned}$$

and:

$$\begin{aligned} \mathrm{minc}(r(x), r(y)) &= \mathrm{minc}\left(\min(255, x + 1), \min(255, y + 1)\right) \\ &= \min\left(\min(255, x + 1), \min(255, \min(255, y + 1) + 1)\right) \\ &= \min(255, x + 1, y + 2) \end{aligned}$$

Considering endomorphisms instead of surjective mappings allows to define the r-operators on finite sets, which is of great importance for implementation purposes, especially for self-stabilizing applications (where the r-mappings need to be strictly increasing [14, 15, 7]).

7.3 Order relation

An idempotent Abelian semi-group $(\mathbb{S}, \oplus)$ admits the order relation $\preceq_\oplus$ defined by $x \preceq_\oplus$ iff $x \oplus y = x$. In this section, we generalize this result to r-semi-groups.

Proposition 47 *Let* $(\mathbb{S}, \triangleleft)$ *be an r-semi-group. Then the following binary relation* $\preccurlyeq_\triangleleft$ *is a partial order relation:*

$$\forall x, y \in \mathbb{S}, \qquad x \preccurlyeq y \Leftrightarrow e_\triangleleft \triangleleft x \triangleleft y = e \triangleleft x$$

Proof. By rank 2 idempotency (Proposition 46), we have: $\forall x \in \mathbb{S}, e_\triangleleft \triangleleft x \triangleleft x = e_\triangleleft \triangleleft x$ and $\preccurlyeq$ is reflexive. Suppose that $x \preccurlyeq y$ and $y \preccurlyeq x$ hold. Then $e_\triangleleft \triangleleft x \triangleleft y = e \triangleleft x$ and $e_\triangleleft \triangleleft y \triangleleft x = e \triangleleft y$. By rank 2 commutativity (Proposition 45), we have $e_\triangleleft \triangleleft y \triangleleft x = e_\triangleleft \triangleleft x \triangleleft y$ and then $e \triangleleft x = e \triangleleft y$. This gives by Proposition 27 (thanks to r-associativity) $r(x) = r(y)$. Since r is injective (Proposition 44, thanks to r-associativity), we have $x = y$ and $\preccurlyeq$ is antisymmetric. Suppose that $x \preccurlyeq y$ and $y \preccurlyeq z$. Then we $e_\triangleleft \triangleleft x = e_\triangleleft \triangleleft x \triangleleft y$ and $e_\triangleleft \triangleleft y \triangleleft z = e_\triangleleft \triangleleft y$ hold. By rank 2 commutativity (Proposition 45), we have:$e_\triangleleft \triangleleft x = e_\triangleleft \triangleleft y \triangleleft x$. By substitution, we have $e_\triangleleft \triangleleft x = e_\triangleleft \triangleleft y \triangleleft z \triangleleft x$. By rank 2 commutativity, we have $e_\triangleleft \triangleleft x = e_\triangleleft \triangleleft x \triangleleft y \triangleleft z$. By substitution, we have $e_\triangleleft \triangleleft x = e_\triangleleft \triangleleft x \triangleleft z$. Then $x \preccurlyeq z$ and $\preccurlyeq$ is transitive. Since $\preccurlyeq$ is reflexive, antisymmetric and transitive, it is a partial order relation. □

7.4 Induced semi-group

In this section, we precise the link between idempotent semi-groups and r-semi-groups. This allows to design new r-operators (see Section 8.5).

Let define the following property:

$$\forall x, y \in \mathbb{S}, \qquad r(x) \triangleleft y \in r(\mathbb{S}) \tag{3}$$

Proposition 48 *Let* $(\mathbb{S}, \triangleleft)$ *be an r-semi-group satisfying Property 3. Then* $(r^2(\mathbb{S}), \oplus)$ *is a magma with* $\oplus$ *defined by* $r^2(x) \oplus r^2(y) = r^2(x) \triangleleft r(y)$.

Proof. For any $x, y \in \mathbb{S}$, we have: $r^2(x) \oplus r^2(y) = r^2(x) \triangleleft r(y)$ by definition of $\oplus$. Then, by homomorphism, we have: $r^2(x) \oplus r^2(y) = r(r(x) \triangleleft y)$. By Property 3, there exists $z \in \mathbb{S}$ such that $r(x) \triangleleft y = r(z)$. Then $r^2(x) \oplus r^2(y) = r^2(z) \in r^2(\mathbb{S})$ and $\oplus$ is an internal composition law on $r^2(\mathbb{S})$. □

In the following, $\oplus$ denotes the internal law defined on $r^2(\mathbb{S})$ by

$$r^2(x) \oplus r^2(y) = r^2(x) \triangleleft r(y) \tag{4}$$

Proposition 49 *Let* $(\mathbb{S}, \triangleleft)$ *be an r-semi-group satisfying Property 3. Then* $\oplus$ *is associative on* $r^2(\mathbb{S})$.

Proof. For any $x, y, z \in \mathbb{S}$, we have by definition of $\oplus$ and homomorphism property of r on $(\mathbb{S}, \triangleleft)$: $r^2(x) \oplus (r^2(y) \oplus r^2(z)) = r^2(x) \oplus (r^2(y \triangleleft r(z)) = r^2(x) \oplus r(r(y) \triangleleft z) = r^2(x) \triangleleft (r(y) \triangleleft z)$. Then, by r-associativity of $\triangleleft$, we have: $r^2(x) \triangleleft (r(y) \triangleleft z) = (r^2(x) \triangleleft r(y)) \triangleleft r(z)$. Then, by definition of r, we have: $(r^2(x) \triangleleft r(y)) \triangleleft (z) = (r^2(x) \oplus r^2(y)) \oplus r^2(z)$. □

Proposition 50 *Let* $(\mathbb{S}, \triangleleft)$ *be an r-semi-group satisfying Property 3. Then* $\oplus$ *is commutative on* $r^2(\mathbb{S})$.

Proof. For any $x, y \in \mathbb{S}$, we have by definition of $\oplus$ and homomorphism property of r on $(\mathbb{S}, \triangleleft)$: $r^2(x) \oplus r^2(y) = r^2(x) \triangleleft r(x) = r(r(x) \triangleleft y)$. Then, by r-commutativity, we have: $r(r(x) \triangleleft y) = r(r(y) \triangleleft x)$. Then, by definition of $\oplus$ and homomorphism property of r, we have: $r(r(y) \triangleleft x) = r^2(y) \triangleleft r(x) = r^2(y) \oplus r^2(x)$. □

Proposition 51 *Let* $(\mathbb{S}, \triangleleft)$ *be an r-semi-group satisfying Property 3. Then* $\oplus$ *is idempotent on* $r^2(\mathbb{S})$.

Proof. For any $x \in \mathbb{S}$, we have by definition of $\oplus$ and homomorphism property of r on $(\mathbb{S}, \triangleleft)$: $r^2(x) \oplus r^2(x) = r^2(x) \triangleleft r(x) = r(r(x) \triangleleft x)$. Then, by r-idempotency, we have: $r(r(x) \triangleleft x) = r(r(x))$. Thus $r^2(x) \oplus r^2(x) = r^2(x)$ and $\oplus$ is idempotent. □

Corollary 52 *Let* $(\mathbb{S}, \triangleleft)$ *be an r-semi-group satisfying* $\forall x, y \in \mathbb{S}, r(x) \triangleleft y \in r(\mathbb{S})$. *Then* $(r^2(\mathbb{S}), \oplus)$ *is a semi-group, where* $\oplus$ *is defined by* $r^2(x) \oplus r^2(y) = r^2(x) \triangleleft r(y)$.

Note that in previous works, the r-operators were defined with a surjective mapping, then bijective by Proposition 44. In this case, if $(\mathbb{S}, \triangleleft)$ is an r-semi-group, then $(\mathbb{S}, \oplus)$ is an idempotent Abelian semi-group with $\oplus$ defined by $x \oplus y = r(x) \oplus y$. The above corollary is more general.

Figure 7 summarizes the properties of the r-operators.

8 Applications of the r-operators

8.1 Related work

The modeling presented in this paper encompasses many situations either in parallel or distributed computing. The results can then be used in a wide range of applications, as explained in further sections. We summarize below related works.

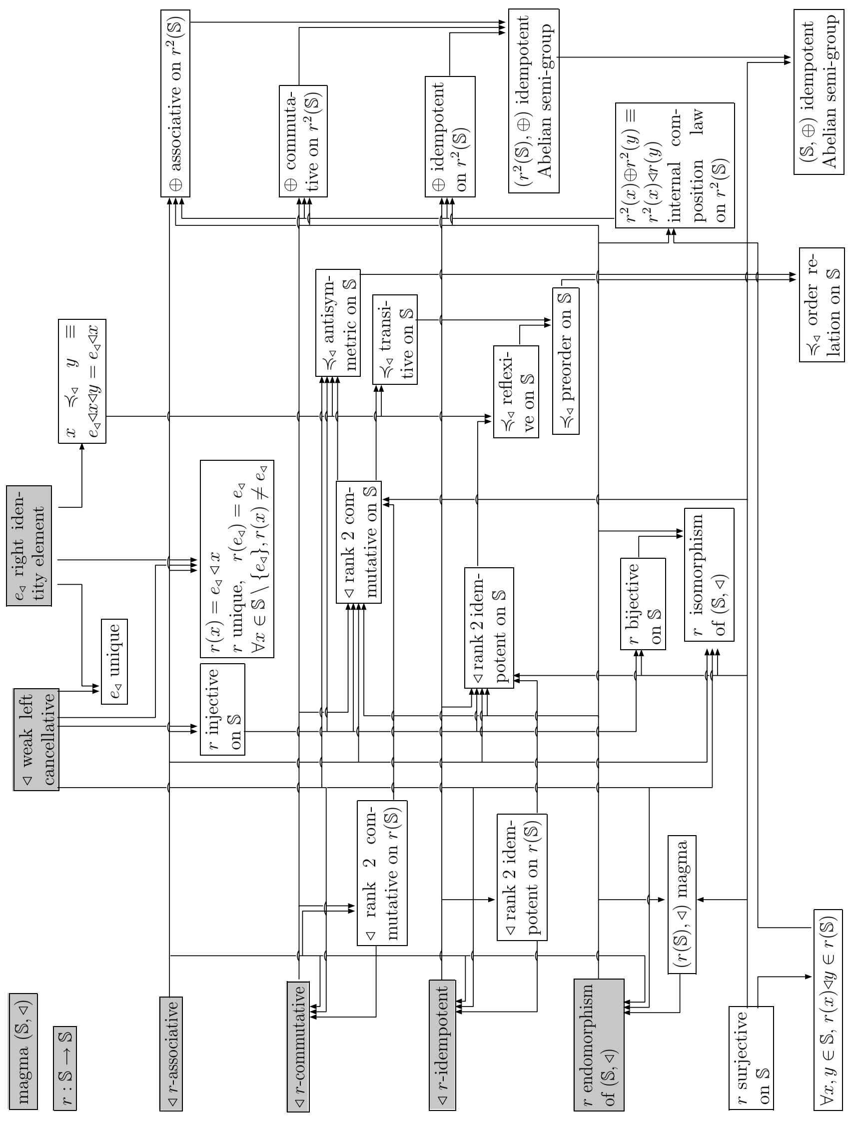

Figure 7: Summary of relationship between properties in an r-semi-group.

In [23], G. Tel showed (in a distributed computing context) that when instantiated by a so-called infimum (that is the law of an idempotent Abelian semi-group, such as the minimum on the integers), the global computation terminates. Note that these operators are associative, commutative and idempotent, and fulfil the requirements related to the wiring and the circuits of the networks. However, the result of a node is simply given by the smallest private data of its ancestors, and this limits the applications.

In [9], this result has been extended to the r-operators (defined with bijective r-mapping), using a proof based on reduction of expressions. Each local computation is assimilated to the building of an expression, composed with the private data of the node, its inputs (considered as expressions) and its local operator. The r-operators have the ability to reduce any such expression to a finite one, ensuring termination of the global computation.

Related works concern path algebra [2, 18], while the path algebra structures require two laws (as in so-called *max-plus* algebra [4]). To the contrary of the r-operators, such algebra do not generalize the idempotent semi-groups but are some particular cases of semi-ring structures (*dioïd*). Other works deal with some relations between algebraic structures and computations on networks (see for instance [5]), but they do not define operators for useful distributed applications as the r-operators do.

8.2 Applications in parallel computing

Using r-operators. A parallel program is a sequential list of parallel operations performed on each processor. Depending on the paradigm, a parallel operation is either a single instruction, a group of instructions or a program. Moreover such operations are more or less coupled, by means of some synchronizations or communications. Generally the less there are such synchronization, the more the speedup is large. Several works have been done in order to reduce the synchronizations in parallel programs.

For instance, in the so-called *asynchronous computations*, the data-dependent constraints are relaxed, and the control of the computation is low. A significant improvement can be obtained, due to the diminution of control (see *e.g.*, [25]).

The modeling proposed in this report applies essentially on data-parallel computations [17] where a parallel operation consists in a single parallel instructions which can easily be modeled with an r-operator. Such a paradigm is often used for image analysis for instance, where a sequence of parallel instruction is applied on a large amount of data (pixels). Thanks to their properties of convergence, the r-operators allow to replace a sequence of several parallel instructions (with a synchronization between each) by a single coarse-grain asynchronous parallel instruction. For instance, instead of using some iterations for computing a distance (which means many synchronizations), a single data-parallel instruction with the r-operator minc can be used. This improves the performances either on hardware or software implementations.

Results. In [10], new parallel asynchronous algorithms have been proposed for image analysis using the r-operators: distance transform, contour closing, image segmentation, mathematical morphology... Some asynchronous hardware implementations have been studied in [16, 22]. An asynchronous communication requires the average communication delay, while a synchronous communication is based on a period, that is computed on the basis of the worst communication delay. Moreover, asynchronism allows lower power consumptions (the clock picks are avoided) and the chips may be smaller (the clock has not to be wired).

A parallel computing library has been developed to perform parallel programs based on r-operators in standard multiprocessors. It relies on some specific C++ classes and on POSIX threads. The r-operators improve the speed-up thanks to some relaxation phenomena and the low number of synchronizations [11, 12, 13].

8.3 Application to distributed computing

Using r-operators. Distributed algorithms resolve either static tasks (*e.g.*, distance computation) or dynamic tasks (*e.g.*, token circulation). The aim of static tasks is to compute a global result, which means that after a running time, processors always produce the same output (*e.g.*, the distance from a source). The specification of a static task is given by the final processor's outputs. To solve such a task, each processor uses a local algorithm. Such a local algorithm builds regularly some new outputs with the data owned by the processor itself and the data it has received from some other nodes. These outputs can then be seen as results of algebraic computations in a set $\mathbb{S}$ to which belong all the data in the distributed system (*e.g.*, the set of integers $\mathbb{N}$ for the distance computation). And the local algorithm can be seen as an algebraic operator, that operates on this set $\mathbb{S}$ (*e.g.*, $\min(x, y+1)$ for the distance computation [14]).

This modeling is not restricted to algorithms that always send the same output to all their neighbors. Indeed, by sending some control information with the output, the ancestors know if they have to use the output or not. This is in fact similar to wireless network, where the same message is sent to all the neighbors, but not necessarily taken into account by each of them, depending on additional information in the header (*e.g.*, address of the receiver).

Note that the local algorithms are not necessarily the same on all the nodes, depending on the distributed algorithm itself, either uniform or not. This means that the operators could be different from one node to another. Note however that our results concerning the r-operators (laws of r-semi-groups) suppose some common properties for the operators, such as defining the same order relation on $\mathbb{S}$ (see Section 7.3).

Note also that this modeling correspond to the asynchronous message passing communication model. The outputs of neighbors are sent asynchronously, and stored locally before being used in the next local computation. The results are then available for stronger models (shared memory, composite atomicity...).

Interest. Modeling a distributed algorithm by means of algebraic computations allows to study the termination of a static task with classical methods, such as fixed points. Moreover the algebraic properties ensuring the termination can simply be checked on the local algorithm (*i.e.*, operator) instead of the whole system. And the results generally apply on a large range of applications by simply varying the operator, leading to generic proofs.

The modeling is also useful to study self-stabilizing algorithms. A distributed algorithm that solves a static task is self-stabilizing if it can reach a legitimate configuration while some transient failures occur [8]. A legitimate configuration is a configuration of the system (state of the processor's memories and communication channels) in which the processor's outputs fulfill the specification of the distributed algorithm. A self-stabilizing algorithm reaches such a configuration in finite time after the transient failures disappear. Self-stabilizing proofs are often long and difficult [19]. A generic modeling is then interesting to study a class of applications instead of a particular one: proofs and properties are shared by several distributed algorithms or applications. Several works deal with generic approaches to prove self-stabilizing properties of distributed algorithms [3, 1, 24, 20, 21]. However none of them rely on an algebraic modeling of the computations, though we think that this method is really generic, as the examples of already developed r-operators show.

Results. In order to use an algebraic modeling of the distributed computations, for studying termination or self-stabilization of static tasks, it is necessary to model the local algorithms by means of operators. The applications of idempotent Abelian semi-groups are limited because a distributed computation with one of these operators gives the same result on all the nodes belonging to the same strongly connected component. Moreover, these operators do not support any transient failures [14]. For instance, in the case of the minimum, if a value smaller than all private data appears in a node due to a transient failure, then it will never be withdrawn, and the output of this node and all its successors will remain illegitimate. In fact, the properties defining the idempotent commutative semi-groups are not necessary for computing on networks. We showed that the r-operators still fulfill the requirements related to wiring and circuits in the network (Requirements 1 and 2 page 5).

In [14], it has been shown that, to the contrary of the infimum, the strictly idempotent r-operators support transient failure. This has been proved in a shared memory model, with the weakest synchronization hypothesis (read-write demon), and assuming that the order relation induced by the operator is total. In [7], this result has been extended to the messages passing networks. Moreover it still holds when there are some fair messages loss, finite messages duplication and arbitrary messages reordering. In [15], a proof with a partial order and a few more constraints in the nodes synchronization (fully distributed demon) has been given using max-plus algebra [4] and fixed-point related techniques. The r-operators permitted to model the distributed computations by means of asynchronous matrix iterations.

From the applications point of view, the r-operators allow to compute: distance related problems (weighted distance, single or multiple source shortest path...), depth-first-search tree computation, ordered list of ancestors, topological sort, diameter computation...

8.4 Using r-operators

In this section, we give some examples of useful r-operators and we summarize the expected properties of the (distributed) algorithms relying on them, before explaining how to build new r-operators.

On numerical sets, the minc operator defined by $\mathrm{minc}(x, y) = \min(x, y + 1)$ solves the distance computation and related problems. By using other r-mappings (such as $x \mapsto x + \mathrm{weight}$), the multiple source shortest path problem is solved similarly [14]. The r-operator $\max(x, y \times \pi)$ solves the best reliable paths related problems (π characterizes the link, $0 < \pi < 1$). The r-operator $\max(x, \min(x, \kappa))$ solves the best capacity paths related problems (κ is the capacity of the link). All these operators lead to self-stabilizing algorithms in unreliable messages passing systems [7]. Note that the operator max (special case of r-operator with the r-mapping identity) solves a silent task on any network, but is not self-stabilizing, except on networks without any circuit. In the same way, the r-operator $\max(x, y + 1)$ leads to a self-stabilizing algorithm only if the network has no circuit (topological sort). The algorithm is not silent if there is a circuit.

With the (non numerical) set of identity's lists, ordered with the lexicographical order, and with the r-mapping that adds to the end of a list the identity of the node, one can build an r-operator that solves the depth-first-search tree [14]. With the set of lists of identity's sets, and with a mapping that shifts a list to the right, one can build an r-operator that gives for each node the ordered list of ancestors [15]. This is useful to discover the k-neighborhood (neighborhood until distance k), or the distance from the farthest node for instance (assuming there is a path between each pair of nodes in the network). To the contrary of the previous operators, this last one defines a partially ordered set, and the self-stabilizing property of the resulting distributed algorithm is established for the fully distributed demon (instead of the read-write demon).

Assuming a synchronous demon, the time complexity of the resulting distributed algorithm is in $O(\mathrm{diameter} + |\mathbb{S}|)$ [14]. The following array summarizes the properties of the distributed algorithm, depending on the algebraic structure.

Abelian semi-group	silent task if there is no circuit	
idempotent Abelian semi-group	silent task, not self-stabilizing	[23], [14]
idempotent r-semi-group	silent task	[9, 14]
strictly idempotent totally ordered r-semi-group	self-stabilizing, read-write demon, shared memory (registers) and unreliable messages passing	[14, 7]
strictly idempotent partially ordered r-semi-group	self-stabilizing, shared memory, fully distributed demon	[15]

8.5 Designing new r-operators

As previously explained, the r-operators have many applications both in parallel and distributed computing. However, the drawback may be the difficulty in designing the r-operators. Based on the results of Section 7.4, we give here some tips to build r-operators. The idea is to start from an idempotent Abelian semi-group, and then to derivate an r-semi-group.

1. Suppose that we seek an r-operator for computing a global task. In order to build the output of a node, a local computation can be considered as discarding some inputs while keeping some others. For the distance computation by example, if a node receives the values 3 from a first neighbor and 5 from a second, it should keep 3 and discards 5. From this choice, we may deduce an order relation $\preccurlyeq$ on the set $\mathbb{S}$ to which belong all the values such that the discarded values are larger than the others (the natural order on the integers for the distance computation). From this order, we may deduce an operator $\oplus$ such that $(\mathbb{S}, \oplus)$ is an idempotent commutative semi-group. For the distance, we find $(\mathbb{N}, \min)$ for instance.

2. If the output of a node is larger than the smallest of its input, then we may deduce an application r that increases its input (*i.e.*, $x \preccurlyeq r(x)$) such that the output of the node is equal to the smallest of its inputs increased by r. In other word, the output would be equal to $x_0 \oplus r(x_1) \oplus \cdots \oplus r(x_p)$ where x_0 is the private data of the node and $x_1 \ldots x_p$ are its inputs. For instance, for the distance computation, if a node receives both 3 and 5, its output (representing its distance from a source in the network) will be equal to 4. We then define $r : \mathbb{N} \to \mathbb{N}$ by $r(x) = x + 1$.

3. On an other hand, if the output of a node is equal to the smallest of its inputs, this may indicate that the silent task is solved with an

operator issued from an idempotent commutative semi-group (*i.e.*, r is the identity mapping $x \mapsto x$). Note that if the output is strictly smaller than its inputs for every nodes, then the global computation may not terminate when there are some circuits in the network: the output will be smaller and smaller.

4. From the results of Section 7.4, we can then define an r-operator using $\oplus$ and r using the relation $r^2(x) \triangleleft r(y) = r^2(x) \oplus r^2(x)$.

The characteristic of both $\preccurlyeq$ and r allow to determine the properties regarding termination and self-stabilization of the related distributed algorithm. For the distance computation, the order $\leq$ is total on $\mathbb{N}$ and $x \mapsto x+1$ is strictly expansive. Hence the r-operator minc leads to a self-stabilizing distributed algorithm even with unreliable messages passing networks.

Obviously, this methodology may not be convenient for all problems. In particular, step 1 gives essentially total order relations. Moreover some complex definition sets $\mathbb{S}$ may be used in order to model a global computation by means of algebraic computation, as shown above with the ordered list of ancestors. Nevertheless, this method allowed to build new r-operators, and then to benefit of generic proofs which is of great important in self-stabilization for instance.

9 Conclusion

In this report, the connection between the local operators and the properties of the global computation on a network has been studied. Modeling the distributed computations by means of algebraic local computations is a convenient way for studying the properties of parallel and distributed calculations.

We first defined a generic modelling where local algorithms are assimilated to algebraic operators. This modelling can be applied to data-parallel computing (often used for image analysis for instance) and distributed static tasks (those which are specified with a global result instead of with a global behavior). We also introduced two important requirements related to the wiring and the circuits of the network.

Then we studied several forms of idempotency (sequence, order) to deal with circuits in the networks. We also presented the weak left cancellation, that permits some simplifications in calculations, even with idempotent operators.

Next, we introduced the rank commutativity and the rank idempotency, which are useful regarding our modelling, where the private data of a node always appear at the first position in the local calculations. Note that rank associativity is not useful. We also showed the link between rank idempotency and order idempotency. Both rank 2 commutativity and rank 2 idempotency are important to fulfill the two requirements related to wiring and circuits.

We then defined the r-properties, that generalize the properties defining idempotent Abelian semi-groups by using an application. Classical properties of idempotency, commutativity and associativity appear as particular cases when the application is the identity mapping $x \mapsto x$. We showed the importance of the r-associativity, the link between r-commutativity and rank 2 commutativity and the link between r-idempotency and rank 2 idempotency.

Using these r-properties, we defined the r-semi-group, generalizing the idempotent Abelian semi-group with an endomorphism. An r-operator is the law of an r-semi-group. The semi-group is a particular case of r-semi-group where the endomorphism is the identity mapping. We showed that an r-semi-group is rank 2 idempotent and rank 2 commutative. Moreover it admits an order relation as the semi-group does. We also showed that an r-semi-group induces an idempotent Abelian semi-group, which is useful to build new r-operators.

Finally, we summarized the application of the r-operators both for parallel and distributed computing. The r-operators can improve the speed-up of some data-parallel computations. A parallel computing library has been designed for exploiting such properties. The r-semi-groups are also particularly useful for proving the self-stabilizing properties of a distributed system. Indeed, the proofs are often complex. When some proofs have been established for a given algebraic structure, it is sufficient to check whether the local algorithm is or not an operator of such a structure instead of writing some potentially long proofs *from scratch*. An important point is that the behavior of the whole distributed system is characterized by some local properties.

The r-operators had already been introduced in previous publications, but partially, with simplistic hypotheses and essentially for specific applications. The main contribution of this report is to present the foundation of the r-operators, and the construction of the r-semi-groups.

Open problems concern the relationship between the partial versus total order relation and the kind of synchronization hypotheses required to ensure the self-stabilization of a system, as well as the completeness of the r-operators for such convergent global computations in networks. Besides these problems, future works may concern the definition of complex recursive r-operators and the development of new applications.

References

[1] Y. Afek and A. Bremler. Self-stabilizing unidirectional network algorithms by power supply. *Chicago Journal of Theoretical Computer Science*, 4(3):1–48, 1998.

[2] A. Aho, J. Hopcropft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Series in Computer Science and Information Processing. Addison-Wesley Publishing, Reading, Ma, USA, 1974.

[3] A. Arora, P. Attie, M. Evangelist, and M. Gouda. Convergence of iteration systems. *Distributed Computing*, 7:43–53, 1993.

[4] F. Baccelli, G. Cohen, G. Olsder, and J.-P. Quadrat. *Synchronisation and Linearity, an algebra for discrete event systems.* Series in Probability and Mathematical Statistics. Wiley, Chichester, UK, 1992.

[5] G. Bilardi and F. Preparata. Characterization of associative operations with prefix circuits of constant depth and linear size. *SIAM Journal of Computing*, 19(2):246–255, April 1990.

[6] N. Bourbaki. *Algèbre.* Éléments de Mathématiques. Hermann, Paris, France, second edition, 1964. Fascicule IV, Livre II.

[7] S. Delaët, B. Ducourthial, and S. Tixeuil. Self-stabilization with r-operators revisited. In *Self-Stabilizing Symposium*, 2005.

[8] S. Dolev. *Self-stabilization.* The MIT Press, 2000.

[9] B. Ducourthial. New operators for computing with *associative nets*. In *Proceedings of the Fifth International Colloquium on Structural Information and Communication Complexity (SIROCCO'98), Amalfi, Italy*, pages 55–65. Carleton Scientific, June 1998.

[10] B. Ducourthial and A. Mérigot. Parallel asynchronous computation for image analysis. *Proceeding IEEE*, 90(7):1218–1229, July 2002.

[11] B. Ducourthial, A. Mérigot, and N. Sicard. Anet: a programming environment for parallel image analysis. In *Proceedings of the IEEE International Workshop on Computer Architectures for Machine Perception (CAMP 2000)*, pages 280–289, Italy, September 2000.

[12] B. Ducourthial, A. Mérigot, and N. Sicard. Parallelizing image analysis algorithms: Anet solutions and performances. In *Proceedings of the IEEE International Workshop on Computer Architectures for Machine Perception (CAMP 2005).* IEEE, 2005.

[13] B. Ducourthial, N. Sicard, and A. Mérigot. Efficient neighborhood-based computations on regions using scans. In *Proceeding of IEEE ICIP 2005*, 2005.

[14] B. Ducourthial and S. Tixeuil. Self-stabilization with r-operators. *Distributed Computing*, 14(3):147–162, July 2001.

[15] B. Ducourthial and S. Tixeuil. Self-stabilization with path algebra. *Theoretical Computer Science*, 1(293):219–236, February 2003. Special issue on max-plus algebra.

[16] D. Dulac, S. Mohammadi, and A. Mérigot. Implementation and evaluation of a parallel architecture using asynchronous communications. In V. Cantoni, L. Lombardi, et al., editors, *Proceedings of the International Workshop on Computer Architectures for Machine Perception (CAMP 1995)*, pages 106–111, Pavia, Italia, September 1995. IEEE Press.

[17] R. Duncan. A survey of parallel computer architectures. *IEEE Computer*, pages 5–16, February 1990.

[18] M. Gondran and M. Minoux. *Graphes et Algorithmes*. Number 37 in Collection de la DER de EDF. Eyrolles, Paris, France, 1979.

[19] M. G. Gouda. The triumph and tribulation of system stabilization. In *Proc. 9th WDAG*, number 72 in LNCS, pages 1–18. Springer Verlag, 1995.

[20] M. G. Gouda and M. Schneider. Maximizable routing metrics. *IEEE/ACM Trans. Netw.*, 11(4):663–675, 2003.

[21] J. Oehlerking, A. Dhama, and O. Theel. Towards automatic convergence verification of self-stabilizing algorithms. In *Seventh International Symposium on Self Stabilizing Systems (SSS'05)*, Barcelona, Spain, October 2005.

[22] G. Privat, F. Robin, M. Renaudin, and B. EL Hassan. A fine-grain asynchronous VLSI cellular array processor architecture. In *Proceedings of the International Symposium on Circuits and Systems*, pages 1041–1044, Seattle, USA, April 1995. IEEE.

[23] G. Tel. *Topics in Distributed Algorithms*, volume 1 of *Cambridge International Series on Parallel Computation*. Cambridge University Press, Cambridge, UK, 1991.

[24] O. Theel. Exploitation of ljapunov theory for verifying self-stabilizing algorithms. In *14th Symposium on Distributed Computing (DISC'00)*, Toledo, Spain, 2000.

[25] A. Üresin and M. Dubois. Effects of asynchronism on the convergence rate of iterative algorithms. *Journal of Parallel and Distributed Computing (JPDC)*, 34:66–81, 1996.