Properties of the r-operators to insure termination of distributed computations

Bertrand Ducourthial
Laboratoire de Recherche en Informatique (bât. 220)
and
Institut d'Electronique Fondamentale (bât. 490)
Université Paris-Sud, 91405 ORSAY Cedex FRANCE
email: ducourth@ief.u-psud.fr

December 15, 1997

Abstract:

In this report, we study the stability of global computations performed in a distributed network of computing entities defined as follows. Each computing entity computes a single value using those it receives from its direct ancestors in the network. This value is then sent to its direct descendants. The network is supplied at the beginning of the global computation with private datum of each entity. The system runs in asynchronous mode. We give some criteria on operators to predict stability of the global computation in whatever topology and using whatever data of the definition set. Hence, when such an operator is used for the global computation, the termination holds a finite time after the beginning of the global computation (note that the termination detection is out of the scope of this work).
We define the class of r-operators which includes previously known s-operators (commutative, associative and idempotent). This new class of operators allows interesting distributed applications.

Keywords: binary operators, wave algorithm, distributed computation, distance in networks

---

Résumé :

Dans ce rapport, nous étudions la stabilité des calculs globaux distribués dans des réseaux d'entités calculantes définies comme suit. Chaque entité calcule une seule valeur à partir de celles qu'elle reçoit de ses ancêtres directs dans le réseau. Cette valeur est ensuite envoyée à ses descendants directs. Le réseau est alimenté au début du calcul global par la donnée privée de chaque entité. Le système est asynchrone.
Nous donnons des critères sur les opérateurs pour prédire la stabilité du calcul global ainsi réalisé sans restriction sur la topologie du réseau ni sur les données de l'ensemble de définition utilisées. Ainsi, lorsqu'un tel opérateur est utilisé pour le calcul global, celui-ci se termine en temps fini (la détection de la terminaison ne rentre pas dans le cadre de ce travail).
Nous définissons la classe des r-opérateurs qui inclue celle des s-opérateurs (associatifs, commutatifs et idempotants). Cette nouvelle classe d'opérateurs autorise d'interessantes applications distribuées.

Mots clés: opérateurs binaires, algorithmes en vagues, calcul distribué, distance dans les réseaux

1 Introduction

In this report, we deal with distributed computing on a simple network of computing entities. We briefly present the model before explain its use for parallel and distributed programming. Then, we summarize related works, and we explain our aims.

The model A computing entity is the smallest unit of material or software that performs computation. Such an entity receives several data and computes a single result with these data and its own datum. We call such a computation a *local computation*. If the result is different from the datum of the entity, it is then sent to neighbor entities. Computing entities are linked together with directed links. The network is the set of inter-entities links. It works in asynchronous mode: sends and receives are not organized in global steps. A global computation begins when the network receives a signal from the outside. It consists of concurrent executions of local computations on the network. It finishes when computations do not affect data in the entities any more. Local computations are defined using binary operators. Each entity uses the same binary operator (uniform global computations). We call such a global computation a *direct–association* since for each computing entity, the data of the direct ancestors are associated to produce its result. This computational model is called Associative Nets.

Associative nets The associative nets model is quite simple. It allows powerful implementations in hardware or software *[Mérigot, 1992, Dulac, 1996]*. We study its ability for high level programming due to $\mathit{direct-association}$s and network management primitives (not presented here). The result of a $\diamond-\mathit{direct-association}$ on a computing entity v is equal, if it exists, to the computation of the binary operator $\diamond$ on all the values received by the computing entity v until stabilization, that is no new datum is received by the entity (see section 3) .

The associative nets model is a powerful parallel computing model *[Mérigot, 1997]*. It is convenient for the design of algorithms *[Mérigot, 1992, Ducourthial, 1995]*, especially in image analysis. Associative nets have been used with binary operators which are commutative, associative and idempotent (in this report, we call such an operator an s-operator). With s-operators, $\mathit{direct-association}$s always terminate in any network initialized with any data. These operators give useful programming primitives. For example, to broadcast a datum from a computing entity v over the whole network, it is sufficient to set to zero all data except in v and then to perform a $\mathbf{max}-\mathit{direct-association}$.

Associative nets have also been used with any kind of operators on directed acyclic graphs, and especially with leaves-to-root oriented trees. Since there is no cycle in these digraphs, all operators can be used and it is possible to perform a $\mathbf{plus}-\mathit{direct-association}$ on integers for example. Using this property, we can compute the sum of the integers in any strongly connected digraph. Indeed, first a leaves-to-root rooted spanning tree is embedded. Then a $\mathbf{plus}-\mathit{direct-association}$ is performed. The sum of all integers of the network is then obtained in the root entity. To broadcast the result of the sum over all the computing entities, a $\mathbf{max}-\mathit{direct-association}$ is used, as explained above.

Related work A *direct − association* can be compared to *diffusing computation*, which is defined as follows [Dijkstra and Scholten, 1980]. The global computation starts when a computing entity receives an external signal from the environment. A computing entity can send messages to its neighbors when it receives an external signal or when it receives a message from one of its neighbor. In our case, we suppose that each computing entity receives an external signal (however a single signal could be used too).

In [Tel, 1994], the concept of *wave algorithm* is developed (similar to that of total algorithms of [Tel, 1991]). A wave algorithm is a global computation which terminates in finite time and such that there is at least one result (decide event) which needs the participation of all computing entities. A *direct − association* is a wave algorithm where each computing entity has to build a result from data of its direct ancestors (note that this corresponds to all the computing entities on an undirected network). Wave algorithms have many applications in distributed computations.

A wave algorithm can be used to broadcast data, as we explained above with a **max**−*direct*−*association*. This technique for broadcast has been proposed in [Segall, 1983] under the name *Propagation of Information (PI)*.

Wave algorithms can be used to compute the infimum of a partially ordered set. Tel showed that $\wedge$, $\vee$, **min**, **max**, **gcd**, **lcm**, $\cap$ and $\cup$ of values local to the computing entities can be computed in a single wave [Tel, 1994, corollary 6.14]. It is due to the fact that these binary operators are associative, commutative and idempotent (s-operators). In section 5, we recall this result with a new proof.

When the binary operator is not idempotent, the global computation needs a leaves-to-root rooted spanning tree, as explain above for the **plus**–*direct*–*association*. This technique is similar to *Propagation Information with Feedback* [Segall, 1983] and to *echo algorithms* [Chang, 1982], which are powerful ways to build distributed algorithms .

Other comparisons could be done with parallel programming paradigms. Indeed, since a *direct − association* with an s-operator gives the same result for each entity of a strongly connected component, this primitive can be compared to *reduction primitives* [Hillis, 1985], that is primitives which give a unique result from an *xector* (vector). Moreover, a *direct − association* on an oriented path is a *prefix computation* [Lakshmivarahan and Dhall, 1994], also called a *scan* [Blelloch, 1990]. On a leaves-to-root oriented tree, it is a more general parallel programming primitive, which has many applications in parallel programming. For instance, in [Hoare et al.,], it is used in an MIMD environment for broadcasting, synchronizing, voting, reductions and asynchronous signal sending under the name of *aggregate functions*.

Aims A global computation such as a *direct*–*association* needs a termination detection system to be used for efficient distributed computations. Such a system is not described in this report. Indeed, we only focus our attention on conditions for local computations to obtain termination after a finite time.

In this report, we study properties of binary operators which allow stability for *direct − association*s in any network, not only acyclic ones. We begin by proving that the idempotency property is a necessary condition to obtain the stability of a global computation. Next, we define r-operators, which are more general than s-operators and we show that distributed computations with idempotent r-operators stabilize in any network. This result extends the use of wave algorithms to r-operators, which can be used for instance to compute distances in networks.

First we give some general graph theory definitions in section 2. In section 3, we define the computational model and in section 4 we detail the purpose of this work. In section 5 we recall results about s-operators. In section 6, we give some necessary conditions for *direct–associations* stabilization. We present r-operators in section 7 and we show their ability for stabilization in section 8. Finally, we give applications using r-operators in section 9.

2 Graph definitions

A *directed graph* or *digraph* $G(V, E)$ is formed with a set of vertices V and a set E of edges (v_1, v_2) which are ordered[1] pairs of vertices of V $(v_1, v_2 \in V)$. The network of computing entities is modeled by a digraph and thus we indifferently speak about network or digraph. A *subgraph* $H(V_H, E_H)$ of a graph $G(V_G, E_G)$ is a graph where $V_H = V_G$ and $E_H \subset E_G$. A *path*, not necessary elementary, from a vertex v_0 to a vertex v_k in a graph $G(V, E)$ is a list of edges of E, $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$. The length of this path is k. The *distance* between two vertices u, v of G, denoted by $\mathrm{d}_G(u, v)$, is the minimum of the lengths of all paths from u to v. The *diameter* of a graph G, denoted by $\mathrm{Diam}(G)$, is the maximum of the distances between all couples of vertices in G. A *cycle* is a path $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ where $v_0 = v_k$. A digraph without any cycle is called a *directed acyclic graph*. The *strongly connected component* of a vertex v is the set of all vertices $w \in V$ such that there exists a path from v to w *and* a path from w to v. G is said strongly connected if it has only one strongly connected component.

The *direct descendants* of a vertex $v \in V$ are all the vertices $w \in V$ such that the edge (v, w) is in E. Their set is denoted by $\Gamma_G(v)$. The *descendants* of v are all the vertices w such that there exists a path from v to w. Their set is denoted by $\widehat{\Gamma_G}(v)$. In the same way, the *direct ancestors* of a vertex $v \in V$ are all the vertices $w \in V$ such that the edge (w, v) is in E. Their set is denoted by $\Gamma_G^{-1}(v)$. The *ancestors* of v are all the vertices w such that there exists a path from w to v. Their set is denoted by $\widehat{\Gamma_G^{-1}}(v)$. The *sources* of a digraph are the vertices s such that there exists a path from s to all other vertices.

A *leaves-to-root rooted spanning tree* of $G(V_G, E_G)$ is a subgraph $H(V_H, E_H)$ of G without any cycle such that there exists a vertex r, called the *root* which is an ancestor of any other vertex.

[1] $(v_1, v_2) \neq (v_2, v_1)$.

3 Computation model

The distributed computing model we consider in this work is a set of identical *computing entities* linked with a *network*. Figure 1 displays such a network with three entities.

A *computing entity* is the smallest unit parcel of material or software that performs a calculation. A computing entity has a limited private memory, $n+1$ incoming buffers called $b_0, b_1, \cdots b_n$ and one outcoming buffer called b_{out}. Order of the incoming buffers is important since the operators are not necessary commutative.

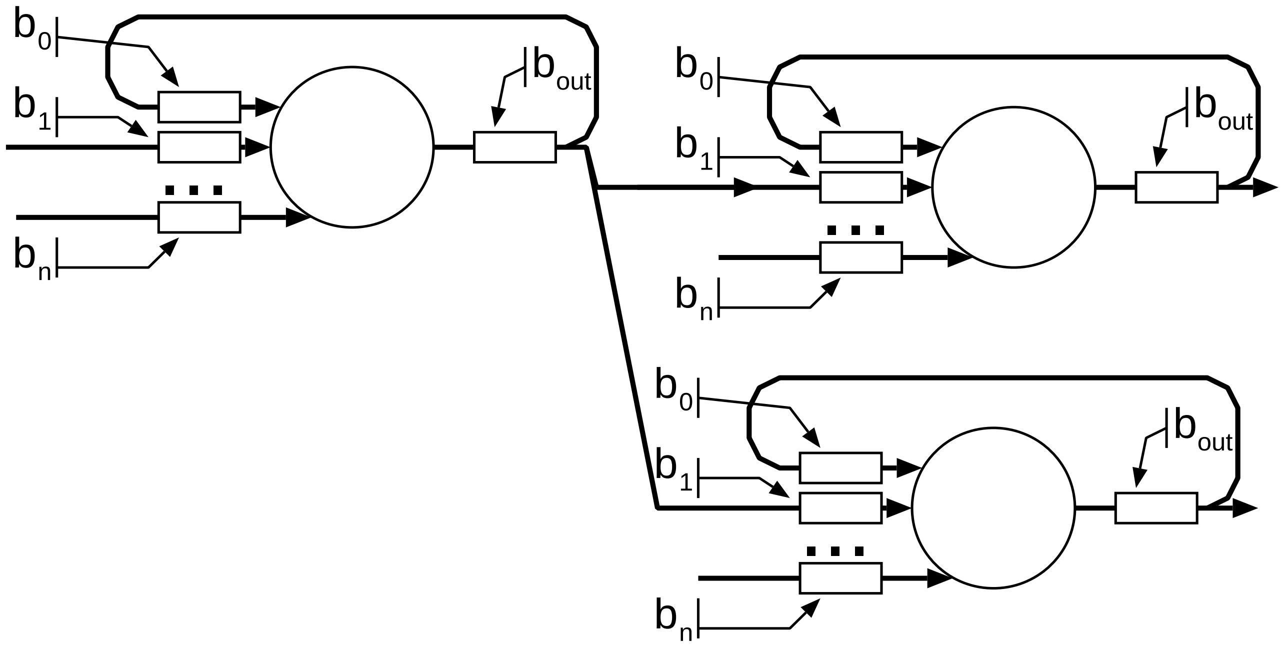

Figure 1: A simple network with three computing entities.

Each link of the network is unidirectional, starts from an outcoming buffer b_{out} of a computing entity and ends at an incoming buffer b_i $(1 \leq i \leq n)$ of another computing entity. Moreover, there is a link from each entity to its own incoming buffer b_0.

Communications are asynchronous which means that the receiver is not waiting for a message, and the sender is not waiting from an acknowledgement of its own message. Moreover the entities are not synchronized. They start a local computation as soon as there is a new data in one of their incoming buffer. As soon as there is a new data in an outcoming buffer, it is sent to all the linked incoming buffers.

It is possible to take into account only a subset of the incoming buffers. In that case, incoming links are broken in front of their incoming buffers. These links or buffers are called *masked*. All buffers but the first (b_0) contain a right-identity element (for the operator of the computing entity, see axiom 3 below) when they are masked (that implies that there exists such a right-identity element). The first buffer contains the initial value of the entity.

At the beginning of a global computation, initial data are in the incoming buffers b_0, and all other incoming links are masked (all the buffers but the first contain a right-identity element). After one local computation, the initial data are also stored in outcoming buffers. Next, incoming links used for the global computation are open and the distributed computation begins. When all buffers are stable, the global computation is ended.

Notice that this model of computing entity allows some variants in $direct-association$s (exclusive-, step-) which are not presented here *[Mérigot, 1992]*.

More formally, we call A the set containing all the data used in the network, and Δ the set of binary operators on A. We assume that the following axioms are verified by the operators of Δ:

• The operators of Δ are internal laws in A:

$$\forall \diamond \in \Delta \quad \forall x, y \in A, \exists ! z \in A \quad \text{such that} \quad x \diamond y = z \tag{1}$$

• The operators of Δ are "left-associative" on A and we use the following convention to simplify the writing of expressions:

$$\forall \diamond \in \Delta \quad (x \diamond y) \diamond z = x \diamond y \diamond z \tag{2}$$

• Each operator of Δ admits a right-identity element:

$$\forall \diamond \in \Delta \quad \exists e_\diamond \in A, \quad \text{such that} \quad \forall x \in A, \quad x \diamond e_\diamond = x \tag{3}$$

• Each operator $\diamond$ of Δ allows left-cancellation, i.e. it defines, for each x in A, an injective mapping $y \mapsto x \diamond y$ from A to A:

$$\begin{gathered} \forall \diamond \in \Delta \quad \forall y, z \in A, \\ (\forall x \in A, \quad x \diamond y = x \diamond z) \quad \Leftrightarrow \quad (y = z) \end{gathered} \tag{4}$$

Axioms 1 and 2 are classical hypotheses. The first one implies that the operators of Δ are deterministic, that is, at any time the same result is obtained from a given pair of elements of A. Axiom 2 is interesting to simplify expression writing. Axiom 3 is needed to allow computing entities with different in-degrees in the network: computing entities with less than n incoming links can be considered by putting e in unused corresponding incoming buffers.

Axiom 4 will help us to characterize operators. The assumption is not strong since all classical operators verify it on their definition set. We can note that all commutative operators on A which have a right-identity element allow left-cancellation on A. It is the same for those having a left-identity element. The left-cancellation property assures that there is only one right-identity element per operator. Indeed suppose that $x = x \diamond e_\diamond = x \diamond e'_\diamond$ for all $x \in A$. Then, using left-cancellation, $e_\diamond = {e'}_\diamond$.

We define an *elementary calculation* as the computation performed by a computing entity v. Such a computation is supposed to be atomic: an elementary computation begins when an incoming buffer is updated with a new value and all new updates of buffers during the computation will be taken into account for the next calculation only. We assume that an elementary computation needs δ_c units of time. We denote by $v_t(b_i)$ the value of the buffer b_i

at the date t. An elementary calculation is then the result of the following operation on the values of the incoming buffers of a computing entity v:

$$v_{t+\delta_c}(b_{\text{out}}) = v_t(b_0) \diamond v_t(b_1) \diamond \cdots \diamond v_t(b_n) \tag{5}$$

Note that all dates used in this report are dates from an external virtual clock, that we use only to study the system. This clock gives the physical time: its value changes even if no event occurs in the network. Such a clock does not exist in the system.

Since we don't use any identity for the computing entities, we are interesting by uniform computation: each entity performs the same elementary calculation. We denote such a global computation by a $\diamond-$*direct*$-$*association* since each entity takes into account its direct ancestors and all the data of the networks are associated to produce the result. Hence a $\diamond-$ *direct*$-$ *association* global computation is performed by concurrent executions of the local computation with the $\diamond$ operator. The result of such a primitive exists only if the global computation terminates.

Originally the model supposes that each link is reliable and has the FIFO-property, since the model was designed for efficient parallel computation. That means that data sent on a link are received exactly once and in the same order that they were sent. However, we can notice that some weaker hypotheses can be considered about network links. In particular the FIFO property is not necessary. Moreover data can be duplicated (several receptions for one sending) on the links without causing any trouble for the global computation. However the modification (garbling) or the creation of a datum is forbidden in general. Since duplication of the sent data is allowed, the loss of messages can be supported by the system with few modifications (for example systematically resending messages after a delay).

If we state that a sent data is received after δ_s units of times, the time complexity of a *direct–association* is at most $\mathrm{Diam}(G)\times(\delta_c+\delta_s)$. However the stabilization can hold more quickly in most cases. For example, with an **and**$-$*direct*$-$*association* on boolean data, the time needed for termination is equal to $d\times(\delta_c+\delta_s)$, where d is the maximum distance between a computing entity with initial datum 1 and another one with initial datum 0 (supposing they exist). Hence simulations have showed that a *direct–association* takes less than $\mathrm{Diam}(G)\times(\delta_c+\delta_s)$ times units in most pratical cases (for example in image analysis *[Dulac, 1996]*). The message complexity is at most $|E|\times\mathrm{Diam}(G)\times(\delta_c+\delta_s)$. However the *direct–association*s are performed on subgraphs (with less edges that the whole network) in most practical cases.

4 Aims

In this report, we study the properties of the operators of Δ needed for all global computations to stabilize on a deterministic result on whatever network with whatever data of A. Deterministic result means that if the same *direct*$-$*association* is performed with the same data at any time, then the same result is always obtained.

It is convenient to consider the $\diamond-$ *direct* $-$ *association* primitive as a mechanism to built regular expressions with brackets, $\diamond$ operators and data of A which were in the network at the beginning of the distributed computing. The result of any global computation is a set of such expressions, one per computing entity. In the following, we will note expressions between square brackets $[\ldots]$, using two wild cards: the notation $[exp]^+$ holds for a non-null number of times the expression $[exp]$, while $[exp]^*$ holds for a number a times could which be null.

As an example, in figure 2 where a $\diamond - \mathsf{direct} - \mathsf{association}$ is performed, we call $E_f(1)$ the final result of the first computing entity and $E_f(2)$ the one of the second entity. f is the first date such that for any date $f' > f$, $E_{f'}(1) = E_f(1)$ and $E_{f'}(2) = E_f(2)$. All masked incoming buffers contain the right-identity element of the operator $\diamond$. At the date f, the $\diamond - \mathsf{direct} - \mathsf{association}$ gives the following expressions: $E_f(1) = [x_0 \diamond e \diamond \cdots \diamond e] = [x_0]$ and $E_f(2) = [y_0[\diamond E_t(2)]^+] = [y_0[\diamond x_0]^+] = [y_0 \diamond x_0 \diamond x_0 \diamond \ldots]$. If this global computation stabilizes, there is a deterministic way to simplify — or reduce — the expressions $E_f(1)$ and $E_f(2)$, using the properties of the operator $\diamond$ on A, such that the expressions above are equal to finished length deterministic expressions.

Figure 2: An example of expressions built with the network: $E_f(1) = [x_0]$ and $E_f(2) = [y_0[\diamond x_0]^+]$.

The use of deterministic operators is not sufficient to obtain deterministic expressions since asynchronicity in the network could lead to different expressions, according to the speed of channels.

More formally, we define the stability of a $\diamond - \mathsf{direct} - \mathsf{association}$ as follows.

Definition 1 *The distributed computation $\diamond - \mathsf{direct} - \mathsf{association}$ with $\diamond$ an operator of Δ* stabilizes on a deterministic result on any network and with any data *if, for any network $G(V, E)$ and any data of A, for any execution performed on this network and with these data, there exists a date t such that for any computing entity v of the network, we have $E_t(v) = E_{t+\tau}(v) = E(v)$, for any $\tau > 0$. In that case, $E(v)$ is called* result on v of the $\diamond - \mathsf{direct} - \mathsf{association}$.

In this report, we develop and study properties of operators allowing to reduce expressions and then to conclude on the stability of the distributed computation.

5 s-operators

In this section we define s-operators and we recall results about termination of distributed computations performed with these operators.

Definition 2 *The operator* $\star \in \Delta$ *is said to be an s-operator on* A *if it is associative, commutative and idempotent on* A*: for all* x*,* y *and* z *in* A*,*

$$\begin{aligned}&(x \star y) \star z = x \star (y \star z)\\&x \star y = y \star x\\&x \star x = x\end{aligned}$$

It can be shown that an s-operator defines a partial ordering. This result is known as the Infimum Theorem, as related in *[Tel, 1991]*. That is why we call such an operator an s-operator: s holds for *sort*. We denote by $\preceq_\star$ the partial ordering relation induced by $\star$ and by $\perp_\star\{x, y\}$ the largest element of the subset of A containing all elements z such that $z \preceq_\star x$ and $z \preceq_\star y$ (supposing it exists, see theorem below).

Theorem 3 (Infimum Theorem) *Let* $\star$ *be a binary associative, commutative and idempotent operator on* A*. Then the binary relation* $\preceq_\star$ *defined by* $(x \preceq_\star y) \equiv (x \star y = x)$ *is a partial ordering on* A*. Moreover* $\perp_\star\{x, y\} = x \star y$*.*

Proof. We must show that $\preceq_\star$ is a transitive, antisymmetric and reflexive binary relation. Suppose $x \preceq_\star y$ and $y \preceq_\star z$. Then $x \star y = x$ and $y \star z = y$. Thus $x \star z = (x \star y) \star z = x \star (y \star z) = x \star y = x$ and $\preceq_\star$ is transitive. Suppose $x \preceq_\star y$ and $y \preceq_\star x$. Then $x \star y = x$ and $y \star x = y$, i.e. $x = y$ and $\preceq_\star$ is antisymmetric. Since $x \star x = x$, we have $x \preceq_\star x$ and $\preceq_\star$ is reflexive. Finally, $(x \star y) \star x = x \star (x \star y) = x \star y$ and $(x \star y) \star y = x \star (y \star y) = x \star y$. Thus $(x \star y) \preceq_\star x$ and $(x \star y) \preceq_\star y$. Now consider $z \in A$ such that $z \preceq_\star x$ and $z \preceq_\star y$. In that case, $z \star x = z$ and $z \star y = z$. Then $z \star (x \star y) = (z \star x) \star y = z \star y = z$. Thus $z \preceq_\star (x \star y)$ and $\perp_\star\{x, y\} = x \star y$. □

Examples of s-operators on $\mathbb{N}$, are given by the minimum $\perp$ ($x \perp y \equiv \mathbf{min}(x, y)$) and maximum $\top$ ($x \top y \equiv \mathbf{max}(x, y)$) operators. For boolean data ($A = \{0, 1\}$), these operators are **and** and **or** respectively. The operators $\cup$, $\cap$, **lcm** (lowest common multiple) and **gcd** (greatest common divisor) are also s-operators on sets or integers.

Let us call $\langle S \rangle_\star$ the set of all the values it is possible to obtain in combining a finite number of data from a subset S of A with the operator $\star$. Since the order induced by an s-operator is not total, we cannot compare any pair of elements. However it is possible to compare combinations of elements and elements themselves. If $\star$ is an s-operator and $S = \{x_1, \cdots, x_n\}$, it is easy to see that $x_1 \star \cdots \star x_n$ is the single element of $\langle S \rangle_\star$ which is smaller than any element of this set (we denote also $\langle S \rangle_\star$ by $\langle x_1, \ldots, x_n \rangle_\star$) with respect to $\preceq_\star$ defined as in theorem 3. We denote such a smallest element by $\perp_\star\langle x_1, \ldots, x_n \rangle_\star$. Moreover, we have $\perp_\star\langle S \rangle_\star \preceq_\star \perp_\star\langle S' \rangle_\star$ for any $S' \subset S$. With these remarks, we can show that if $\star$ is an s-operator, $\star-$*direct*$-$*association* will always stabilize in whatever network with whatever data. This result is similar to the one of *[Tel, 1994, theorem 6.11]*. However we prove it with a different method.

Theorem 4 *If* $\star \in \Delta$ *is an s-operator on* A*, then the global computation* $\star-$*direct*$-$*association will stabilize on a deterministic result in whatever network* G *using whatever data of* A*, such that the result on each computing entity* v *is* $\perp_\star\langle \widehat{\Gamma_G^{-1}}(v) \rangle_\star$*.*

Proof. Let us call $G(V, E)$ the network. All intermediate results produced by the computing entities are smaller than their incoming data. Hence data leaving any entity are smaller than those coming to the entity. This means that all data in transit in the network are smaller and smaller. Intermediate results on each entity keep getting smaller and stabilize on the smallest value it is possible to built with data received during the global computation, that is $\perp_\star\langle\widehat{\Gamma_G^{-1}}(v)\rangle_\star$. □

Notice that after a $\star-$*direct*$-$*association* the same result is obtained on each entity of the same strongly connected component of the network.

Before defining r-operators and extend the result of theorem 4, we give a necessary condition for stabilization.

6 Necessary condition

In this section, we present a necessary condition to obtain the $\diamond-$*direct*$-$*association* stabilization on a deterministic result, for a given operator $\diamond$ in Δ. We use the notation $x^0 = e_\diamond$ and, for $k \geq 1$, $x^k = \underbrace{x \diamond \cdots \diamond x}_{k}$.

Definition 5 *The operator $\diamond \in \Delta$ is said convergent on A if for any $x \in A$, there exists $l \in \langle x\rangle_\diamond$ such that $l \diamond x = x \diamond l = l$.*

A convergent operator has, for any subset of A, at least one limit l which can be reached from combination of elements of the subset. When any subset of A admits the same limit l, it is an absorbative element on A ($\forall x \in A$, $l \diamond x = x \diamond l = l$).

Expressions built with such an operator can always be reduced and then a $\diamond-$*direct*$-$*association* will stabilize on any network with any data of A.

However, there exist some operators which are not convergent but allow stabilization.

Definition 6 *The operator $\diamond$ of Δ is idempotent of order $p \in \mathbb{N} \setminus \{0\}$ on A if for any $x \in A$, $x^p = x^{p+q}$ for any $q \in \mathbb{N}$.*

Proposition 7 *If the operator $\diamond \in \Delta$ is not idempotent on A for any order $p \in \mathbb{N} \setminus \{0\}$, then the $\diamond-$direct$-$association cannot stabilize on whatever network with whatever data of A.*

Proof. Consider the network of figure 3. The result of the computation is $[x]^+$. Then $\diamond-$*direct*$-$*association* can stabilize with any datum $x \in A$ only if $[x]^+$ can be reduced for each $x \in A$. This can be done only if there exists $p \in \mathbb{N}$ such that $\diamond$ is idempotent of order p. □

Note that idempotent operators are convergent.

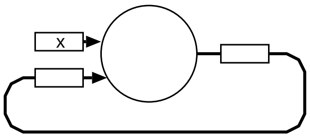

Figure 3: Idempotency of order $p \in \mathbb{N} \setminus \{0\}$ is necessary.

7 r-operators

Results about stability can be obtained with weaker hypotheses than those of s-operators. In this section, we introduce r-operators. Let us first define three new properties on binary operators.

7.1 Definitions

Definition 8 *A binary operator $\diamond$ of Δ is said r-associative on A if there exists a mapping $r : A \to A$ such that for all x, y and z in A,*

$$(x \diamond y) \diamond r(z) = x \diamond y \diamond r(z) = x \diamond (y \diamond z)$$

For instance, the operator $\oplus$ defined on $A = \mathbb{N}$ by $x \oplus y \equiv x + 2y$ is r-associative with the mapping $r(z) = 2z$ from $\mathbb{N}$ to $\mathbb{N}$. Indeed $(x \oplus y) \oplus r(z) = x + 2y + 2r(z)$ while $x \oplus (y \oplus z) = x \oplus (y + 2z) = x + 2y + 4z$. An associative operator is **Id**-associative with the identity mapping $\mathbf{Id} : x \mapsto x$.

Definition 9 *The operator $\diamond$ of Δ is said r-commutative on A with the mapping r from A to A if for all x, y in A, we have:*

$$r(x) \diamond y = r(y) \diamond x$$

For instance, the subtraction operator on $\mathbb{Z}$ is **op**-commutative with the mapping **op** from $\mathbb{Z}$ to $\mathbb{Z}$ defined by $x \mapsto -x$. Indeed $\mathbf{op}(x) - y = -x - y = \mathbf{op}(y) - x$. The following proposition is immediate and it will help us to prove other results.

Proposition 10 *If $\diamond$ in Δ is r-commutative on A with r a bijective mapping from A to A, then for all x, y in A, the following relations are equivalent:*

$$\begin{array}{lcl} r(x) \diamond y & = & r(y) \diamond x \\ x \diamond y & = & r(y) \diamond r^{-1}(x) \\ x \diamond r^{-1}(y) & = & y \diamond r^{-1}(x) \end{array}$$

Definition 11 *A binary operator $\diamond$ of Δ is said r-idempotent with r a mapping from A to A, if for all x in A,*

$$r(x) \diamond x = r(x)$$

For instance, the operator $\mathbf{p}$, defined on $\mathbb{N}$ by $x\mathbf{p}y = 0$ if x and y have the same parity and 1 if not, is r-idempotent with the mapping r defined on $\mathbb{N}$ by $r(x) = 0$ if x is even and $r(x) = 1$ if x is odd. As above, the following proposition is immediate and will help us in more complex proofs.

Proposition 12 *If the operator $\diamond$ of Δ is r-idempotent with r a bijective mapping from A to A, then for all x of A, $x \diamond r^{-1}(x) = x$.*

We now define r-operators in a similar way as s-operators were defined (definition 2).

Definition 13 *The operator $\blacktriangledown \in \Delta$ is an r-operator on A if there exists a bijective mapping r from A to A such that $\blacktriangledown$ is r-associative, r-commutative and r-idempotent on A:*

$$\begin{aligned}
&(x \blacktriangledown y) \blacktriangledown r(z) = x \blacktriangledown (y \blacktriangledown z)\\
&r(x) \blacktriangledown y = r(y) \blacktriangledown x\\
&r(x) \blacktriangledown x = r(x)
\end{aligned}$$

Notice that we always call such an operator an r-operator, even if the mapping is not called r (r holds for *reduce* in the term r-operator). It can be shown using r-associativity and left-cancellation property (axiom 4) that if an r-operator is defined using only a surjective mapping r, then this mapping is necessary bijective (theorem 25). Moreover, some results in this report proved with fewer hypotheses (not all the three of r-operators) are still true if r is only surjective. In that case, we must build a reverse application from the surjective mapping r, by choosing an element in each set $\{x \in A, r(x) = y\}$. However, to simplify, we state that r is bijective even for these results.

To conclude this section, we define a fourth property of binary operators.

Definition 14 *A binary operator $\diamond$ of Δ is said two-commutative on A if for all x, y and z in A,*

$$x \diamond y \diamond z = x \diamond z \diamond y$$

For instance the operator $\oplus$ defined above is two-commutative but not commutative. A commutative and associative operator is two-commutative: $x \diamond y \diamond z = x \diamond (y \diamond z) = x \diamond z \diamond y$. Note that the two-commutativity property (i.e. commutativity from rank two in expressions) is equivalent to commutativity from any greater rank ($x_1 \diamond \ldots \diamond x_{k-1} \diamond x_k \diamond xk + 1 = x_1 \diamond \ldots \diamond x_{k-1} \diamond x_{k+1} \diamond x_k$, with $k > 2$), using the right-identity element.

7.2 Properties of r-operators

In the following, we suppose that r is the bijective mapping associated to the r-operator $\blacktriangledown$ of Δ with e the right-identity element of $\blacktriangledown$. We use the following notation for iterations of r: $r^0(x) = x$ and $r^k(x) = \underbrace{r(r(\cdots r(}_{k} x \underbrace{)\cdots))}_{k}$.

Proposition 15 *The mapping r verifies $r(x) = e \blacktriangledown x$ for all x in A.*

Proof. For all $x, y \in A$, $x \blacktriangledown r(y) = x \blacktriangledown e \blacktriangledown r(y) = x \blacktriangledown (e \blacktriangledown y)$ (r-associativity). Thus $r(y) = e \blacktriangledown y$ using left-cancellation. □

Proposition 16 *The kernel of r, $\mathrm{Ker}(r) = \{t \in A, r(t) = e\}$ is the singleton $\{e\}$.*

Proof. Let t be in $\mathrm{Ker}(r)$. For all $x, y \in A$, $x \blacktriangledown y = x \blacktriangledown y \blacktriangledown r(t) = x \blacktriangledown (y \blacktriangledown t)$ (r-associativity). Then using axiom 4 (left-cancellation), $y = y \blacktriangledown t$. Since $y = y \blacktriangledown e$, we find $e = t$ using left-cancellation of $\blacktriangledown \in \Delta$ again. □

A consequence of proposition 16 is that there is one and only one element t in A such that $r(t) = e$.

Proposition 17 *The bijective mapping r is an isomorphism of $(A, \blacktriangledown)$, where $\blacktriangledown$ is an r-operator on A.*

Proof. For all $x, y, u, v \in A$, $x \blacktriangledown y \blacktriangledown r(u) \blacktriangledown r^2(v) = (x \blacktriangledown y) \blacktriangledown (r(u) \blacktriangledown r(v))$ (r-associativity). Since r is surjective, there exists $t \in A$ such that $r(t) = r(u) \blacktriangledown r(v) \in A$. Then, $x \blacktriangledown y \blacktriangledown r(u) \blacktriangledown r^2(v) = x \blacktriangledown y \blacktriangledown r(t) = x \blacktriangledown (y \blacktriangledown t)$ (r-associativity).

In an other hand, $x \blacktriangledown y \blacktriangledown r(u) \blacktriangledown r^2(v) = x \blacktriangledown (y \blacktriangledown u) \blacktriangledown r^2(v) = x \blacktriangledown ((y \blacktriangledown u) \blacktriangledown r(v)) = x \blacktriangledown (y \blacktriangledown (u \blacktriangledown v))$ (r-associativity).

Then, we have $x \blacktriangledown y \blacktriangledown r(u) \blacktriangledown r^2(v) = x \blacktriangledown (y \blacktriangledown t) = x \blacktriangledown (y \blacktriangledown (u \blacktriangledown v))$, and using left-cancellation, we find $t = u \blacktriangledown v$. Thus, $r(u \blacktriangledown v) = r(t) = r(u) \blacktriangledown r(v)$ and r is a homomorphism and therefore an isomorphism of $(A, \blacktriangledown)$ (since r is a bijective mapping from A to A). □

Proposition 18 *If $\diamond \in \Delta$ is r-associative with r a bijective mapping from A to A, then if $\diamond$ is two-commutative, it is r-commutative, and vice-versa.*

Proof. Suppose the operator $\diamond \in \Delta$ is r-associative and r-commutative on A. Then, for all $x, y, z \in A$, $x \diamond y \diamond z = x \diamond (y \diamond r^{-1}(z)) = x \diamond (z \diamond r^{-1}(y))$ (proposition 10). Thus $x \diamond y \diamond z = x \diamond z \diamond y$ and $\diamond$ is two-commutative.

Now suppose $\diamond$ is r-associative and two-commutative. Then for all $x, y, z \in A$, we have $x \diamond y \diamond z = x \diamond z \diamond y$. Thus $x \diamond (y \diamond r^{-1}(z)) = x \diamond (z \diamond r^{-1}(y))$. Hence, using left-cancellation, we obtain $y \diamond r^{-1}(z) = z \diamond r^{-1}(y)$ and $\diamond$ is r-associative (proposition 10). □

Proposition 19 *If $\diamond \in \Delta$ is r-associative and r-idempotent on A with r a bijective mapping from A to A, then $\diamond$ is idempotent of order two and verifies for all $x, y \in A$, $x \diamond y \diamond y = x \diamond y$.*

Proof. For all $x \in A$, we have $x \diamond x \diamond x = x \diamond (x \diamond r^{-1}(x)) = x \diamond x$ (r-idempotency, proposition 12). Similarly, for all $x, y \in A$, $x \diamond y \diamond y = x \diamond (y \diamond r^{-1}(y)) = x \diamond y$. □

7.3 Associated s-operator

In this section we show that there exists an s-operator associated to each r-operator.

Theorem 20 *If $\blacktriangledown$ is an r-operator on A with a bijective mapping r from A to A, then the operator $\star$ defined by $x \star y = x \blacktriangledown r^{-1}(y)$ is in Δ and is an s-operator on A.*

Proof. The operator $\star$ is associative on A: for all $x, y, z \in A$, $x \star (y \star z) = x \blacktriangledown r^{-1}(y \star z) = x \blacktriangledown r^{-1}(y \blacktriangledown r^{-1}(z)) = x \blacktriangledown (r^{-1}(y) \blacktriangledown r^{-2}(z)) = x \blacktriangledown r^{-1}(y) \blacktriangledown r^{-1}(z)$ (r is an isomorphism of $(A, \blacktriangledown)$, proposition 17). But $x \blacktriangledown r^{-1}(y) \blacktriangledown r^{-1}(z) = (x \star y) \blacktriangledown r^{-1}(z) = (x \star y) \star z$. Therefore $\star$ is associative.

For all $x, y \in A$, $x \star y = x \blacktriangledown r^{-1}(y) = y \blacktriangledown r^{-1}(x)$ (r-commutativity of $\blacktriangledown$, proposition 10). Thus $\star$ is commutative on A.

Since $\blacktriangledown$ is r-idempotent and r is a bijective mapping, $x \blacktriangledown r^{-1}(x) = x$ (proposition 12). Thus $x \star x = x$, and $\star$ is idempotent.

Therefore, the operator $\star$ is an s-operator on A.

It is an internal law on A because it is built with an internal law ($\blacktriangledown$) and a bijective mapping (r) from A to A. Its right-identity element is the same as that of $\blacktriangledown$ (called e): $x \star e = x \blacktriangledown r^{-1}(e) = x$ (see proposition 16). Since $\star$ has a right-identity element and is commutative, it allows left-cancellation. Hence $\star \in \Delta$. □

In the following, $\star$ is the s-operator associated to the r-operator $\blacktriangledown$ by $x \star y = x \blacktriangledown r^{-1}(y)$. We give some properties of the operator $\star$.

Proposition 21 *The s-operator $\star$ is r-commutative.*

Proof. $r(x) \star y = r(x) \blacktriangledown r^{-1}(y) = x \blacktriangledown y$ ($\blacktriangledown$ is r-commutative). $r(x) \star y = x \blacktriangledown y = x \star r(y) = r(y) \star x$ ($\star$ is commutative). Then $r(x) \star y = r(y) \star x$ and $\star$ is r-commutative (proposition 10). □

Proposition 22 *The mapping r is an isomorphism of $(A, \star)$.*

Proof. $r(x \star y) = r(x \blacktriangledown r^{-1}(y)) = r(x) \blacktriangledown y = r(x) \star r(y)$ (proposition 17). □

7.4 Idempotent r-operators

An idempotent r-operator has interesting properties.

Proposition 23 *If $\blacktriangledown$ is an idempotent r-operator, then its associated s-operator $\star$ verifies $x \preceq_\star r^k(x)$ for any $k \in \mathbb{N}$, where $\preceq_\star$ is the partial order on A defined by $x \preceq_\star y$ if and only if $x \star y = x$.*

Proof. Theorem 3 shows that $x \star y = x$ defines a partial order $\preceq_\star$. The proposition is true for $k = 0$, even if $\blacktriangledown$ is not idempotent: $x \star x = x$ since $\star$ is an s-operator. Then $x \preceq_\star x$.

For $k > 0$, we prove the proposition by recurrence on k, using the idempotency property of $\blacktriangledown$. Since $\blacktriangledown$ is idempotent, for all $x \in A$, $x \blacktriangledown x = x$. Thus $x \star r(x) = x$ (theorem 20) and $x \preceq r(x)$. Suppose now that $x \preceq r^{k-1}(x)$. Then $x \star r^k(x) = (x \star r^{k-1}(x)) \star r^k(x) = x \star (r^{k-1}(x) \star r^k(x)) = x \star r^{k-1}(x \star r(x)) = x \star r^{k-1}(x \blacktriangledown x) = x \star r^{k-1}(x) = x$. Hence $x \preceq_\star r^k(x)$ for any $k \in \mathbb{N}$ (recurrence hypothesis). □

Proposition 24 *If $\blacktriangledown$ is an idempotent r-operator, then it is r^{-1}-idempotent. Moreover its associated s-operator $\star$ is r^{-1}-idempotent too.*

Proof. Following proposition 23, $x \preceq_\star r^2(x)$. That is: $x \star r^2(x) = x$ i.e. $x \blacktriangledown r(x) = x$ and thus $r^{-1}(x) \blacktriangledown x = r^{-1}(x)$ (proposition 17).

Following theorem 20, $x \blacktriangledown x = x$ is equivalent to $x \star r(x) = x$ and to $r^{-1}(x) \star x = r^{-1}(x)$ (proposition 22). □

7.5 How to build r-operators

In the previous section we explained how to obtain an s-operator from an r-operator using axioms of Δ. In this section, we summarize relations between r-operators of Δ and s-operators to facilitate the construction of r-operators from known s-operators.

Theorem 25 *If for all $x, y \in A$, the binary operators $\blacktriangledown$ and $\star$ verify $x \blacktriangledown y = x \star r(y)$ with r a mapping from A to A then the three following propositions are equivalent:*

(1) r is a surjective mapping from A to A, the operator $\blacktriangledown$ has a right-identity element, it allows left-cancellation and it is r-commutative, r-associative and r-idempotent on A,

(2) $\blacktriangledown$ is an r-operator defined on A which allows left-cancellation and admits a right-identity element,

(3) $\star$ is an s-operator on A admitting an identity element and $r : A \to A$ is an isomorphism of $(A, \star)$.

Proof. (1) $\Rightarrow$ (2) We only need to prove that r is injective. For all $x, y, a, b \in A$, $r(x) = r(y)$ implies $a \blacktriangledown b \blacktriangledown r(x) = a \blacktriangledown b \blacktriangledown r(y)$. Then $a \blacktriangledown (b \blacktriangledown x) = a \blacktriangledown (b \blacktriangledown y)$ (r-associativity) and thus, using left-cancellation, $x = y$.

(2) $\Rightarrow$ (3) If $\blacktriangledown$ is a binary operator allowing left-cancellation and admitting a right-identity element, then it is in Δ. Thus, $\star$ is an s-operator on A (theorem 20) and r is an isomorphism of $(A, \star)$ (proposition 17).

(3) $\Rightarrow$ (1) If r is an isomorphism of $(A, \star)$, it is a surjective mapping on A, and for all $x, y, z \in A$: $(x \blacktriangledown y) \blacktriangledown r(z) = x \star r(y) \star r^2(z) = x \star (r(y) \blacktriangledown r(z)) = x \blacktriangledown r^{-1}(r(y) \blacktriangledown r(z)) = x \blacktriangledown (y \blacktriangledown z)$

and thus $\blacktriangledown$ is r-associative; $x \blacktriangledown r^{-1}(y) = x \star y = y \star x = y \blacktriangledown r^{-1}(x)$ and $\blacktriangledown$ is r-commutative (proposition 10); $r(x) \blacktriangledown x = r(x) \star r^{-1}(x) = x$ (propositions 21 and 10) and thus $\blacktriangledown$ is r-idempotent. Moreover if $x \blacktriangledown y = x \blacktriangledown z$, $x \star r(y) = x \star r(z)$ and thus $y = z$ ($\star$ allows left-cancellation and r is bijective). Thus $\blacktriangledown$ allows left-cancellation on A. Finally, if $e \in A$ is the identity element of $\star$, $x \blacktriangledown r^{-1}(e) = x \star e = x$ and $r^{-1}(e)$ is a right-identity element of $\blacktriangledown$. $\square$

Hence, for each s-operator $\star$ on A, we can build as many r-operators $\blacktriangledown$ as there are isomorphisms on $(A, \star)$. Moreover, to obtain an idempotent r-operator, the isomorphism r must verify $x \preceq_\star r(x)$ or, equivalently, $\star$ must be (r^{-1})-idempotent: $x \preceq_\star r(x) \Leftrightarrow x \star r(x) = x \Leftrightarrow r^{-1}(x) \star x = r^{-1}(x)$. In section 9 we give an example of idempotent r-operator.

8 Sufficient conditions

We show in this section that $\blacktriangledown-\mathit{direct}-\mathit{association}$ stabilizes if $\blacktriangledown$ is an idempotent r-operator. The idempotency property is necessary, as we showed in section 6.

The idea is, for each computing entity, to reduce expressions in suppressing "repeated terms" to obtain a final expression which is always the same for all further executions of the global computation on the same data. We define a repeated term as follows. We can consider that computing entities do not transform received data into new ones but build expressions with received expressions. Hence, any expression in the network is built with initial data only. Finally, we say that there is a repeated term in an expression when the same initial datum appears several times in this expression. For instance, in the following expression, y and z are repeated terms: $[x \diamond (y \diamond z) \diamond r(z) \diamond (r^k(z) \diamond y))]$.

Theorem 4, which deals with s-operators $\star$, could have been proved following this method. Each entity $v \in V$ will receive all the initial values of its ancestors during the global computation, inside some expressions eventually with brackets, depending on the topology of the network. Since $\star$ is associative, the resulting expression in v can be rewritten suppressing brackets. Next, since $\star$ is commutative, it is possible to group identical data together, and using associativity again, it is possible to isolate them inside brackets. Finally, since $\star$ is idempotent, it is possible to suppress repeated terms inside brackets, and then these brackets containing now a single value can be deleted. The resulting expression on v becomes then a finished list of data separated by the operator $\star$: $[a_0 \star \cdots \star a_k]$, where data $a_0, \ldots, a_k$ are initial data of computing entities which are ancestors of v, that is data that v has received. Hence, the result in v is $\perp_\star \langle \widehat{\Gamma_G^{-1}(v)} \rangle_\star = a_0 \star \cdots \star a_k$.

We apply the same method for idempotent r-operators. To prove that global computations stabilize with any idempotent r-operator of Δ, we need the following lemma.

Lemma 26 *If $\blacktriangledown \in \Delta$ is an idempotent r-operator, then for all $a, x \in A$, for all $k \in \mathbb{N} \setminus \{0\}$,*

$$\begin{array}{ll}
x \blacktriangledown x = x & (1) \\
x \blacktriangledown r^k(x) = x & (2) \\
r^k(x) \blacktriangledown x = r(x) & (3) \\
a \blacktriangledown x \blacktriangledown x = a \blacktriangledown x & (4) \\
a \blacktriangledown r^k(x) \blacktriangledown x = a \blacktriangledown x & (5) \\
a \blacktriangledown x \blacktriangledown r^k(x) = a \blacktriangledown x & (6) \\
x \blacktriangledown r^{-k}(x) = r^{1-k}(x) & (7) \\
r^{-k}(x) \blacktriangledown x = r^{-k}(x) & (8) \\
a \blacktriangledown r^{-k}(x) \blacktriangledown x = a \blacktriangledown r^{-k}(x) & (9) \\
a \blacktriangledown x \blacktriangledown r^{-k}(x) = a \blacktriangledown r^{-k}(x) & (10)
\end{array}$$

Proof. The first equation is due to the idempotency hypothesis. The second one is equivalent to $x \preceq_\star r^{k+1}(x)$ (theorems 20 and 3), which is true for idempotent r-operators (proposition 23). The third equation is proved using the second one and the r-commutativity (see proposition 10): $r^k(x) \blacktriangledown x = r(r^{k-1}(x)) \blacktriangledown x = r(x) \blacktriangledown r^{k-1}(x) = r(x)$. The fourth equation is due to idempotency of order two of $\blacktriangledown$ (proposition 19). The fifth equation is proved using the r-associativity, proposition 17 and the third equation: $a \blacktriangledown r^k(x) \blacktriangledown x = a \blacktriangledown (r^k(x) \blacktriangledown r^{-1}(x)) = a \blacktriangledown r^{-1}(r^{k+1}(x) \blacktriangledown x) = a \blacktriangledown r^{-1}(r(x)) = a \blacktriangledown x$. The sixth equation is due to the two-commutativity (proposition 18) and the fifth equation. The seventh equation is proved as follows: $r^k(x \blacktriangledown r^{-k}(x)) = r^k(x) \blacktriangledown x = r(x)$ (proposition 17 and equation 3). Thus $x \blacktriangledown r^{-k} = r^{1-k}(x)$. The eighth equation is proved similarly. For the ninth equation, we use the r-associativity, the isomorphism property of r and the second equation: $a \blacktriangledown r^{-k}(x) \blacktriangledown x = a \blacktriangledown (r^{-k}(x) \blacktriangledown r^{-1}(x)) = a \blacktriangledown r^{-k}(x \blacktriangledown r^{k-1}(x)) = a \blacktriangledown r^{-k}(x)$. Finally, the tenth equation is equivalent to the ninth, using the two-commutativity (proposition 18). □

We use the following terminology. An *r-expression* is an expression written with the r-operator $\blacktriangledown$. An *s-expression* is an expression written with the s operator $\star$. Any r-expression is equivalent to an s-expression using the s-operator $\star$ associated to the r-operator $\blacktriangledown$ (theorem 20). The r-expression at date t on a computing entity v is denoted by $E_t^{\blacktriangledown}(v)$, while the equivalent s-expression is denoted by $E_t^{\star}(v)$. When the operator is not important, we simply use $E_t(v)$. We can consider that there are $n+1$ terms in any expression since these expressions are built by computing entities in combining values of their $n+1$ incoming buffers. These $n+1$ terms can be expressions themselves.

Lemma 26 indicates that any repeated term can be suppressed in any expression without brackets formed with an idempotent r-operator. Moreover, except for the first datum of the expression, the smallest power of r of each repeated term is kept. In fact, it is easy to see that all expressions in buffer b_{out} of the computing entity u begin with x_u, the initial datum of u without any power of r. Moreover, if an expression begins with x_u (without power of r), then this expression has been produced by u (except if some computing entities have started the global computation with the same initial data). Indeed, at each time a datum is sent, it is directly incorporated into the r-expression of the receiver entity, and then incorporated under a power of r in the corresponding s-expression (theorem 20).

Lemma 27 *If $\blacktriangledown \in \Delta$ is an r-operator and if the network $G(V,E)$ is initialized with $|V|$ distinct data, then there exists a date t such that for all $u, v \in V$, the two following propositions are equivalent:*

(1) $\mathrm{d}_G(u,v) = k$

(2) the expression $E^\star_t(v)$ obtained with $\blacktriangledown$−**direct−association** *contains $r^k(x_u)$, where x_u is the initial datum of u, and k is the smallest power of $r(x_u)$ in the expression.*

Proof. Notice that since the network is initialized with $|V|$ distinct data, no two computing entities have the same initial datum.

We prove the lemma by recurrence on the distance k. It is true for $k = 0$. Indeed, only the expression on u begins with x_u.

Suppose now that the lemma is true for all $k' < k$. Let u, v, w be three entities in G such that $\mathrm{d}_G(u,v) = k-1$, $\mathrm{d}_G(u,w) = k$ and $\mathrm{d}_G(v,w) = 1$. There exists a date t such that the s-expression in v is $E^\star_t(v) = [x_v \star y_1 \star \cdots \star y_{i-1} \star r^{k-1}(x_u) \star y_{i+1} \star \cdots y_n]$ for some $i \in \{1, \ldots, n\}$. Since w is a direct descendant of v in G, there exists a date $t' > t$ such that $E_{t'}(w) = [x_w \blacktriangledown z_1 \cdots \blacktriangledown z_{j-1} \blacktriangledown (x_v \star y_1 \star \cdots \star y_{i-1} \star r^{k-1}(x_u) \star y_{i+1} \star \cdots y_n) \blacktriangledown z_{j+1} \blacktriangledown \cdots \blacktriangledown z_n]$ for some $j \in \{1, \ldots, n\}$. Using theorem 20 and proposition 22, we have $E_{t'}(w) \equiv E^\star_{t'}(w) = [x_w \star r(z_1) \star \cdots \star r(z_{j-1}) \star r(x_v) \star r(y_1) \star \cdots \star r(y_{i-1}) \star r^k(x_u) \star r(y_{i+1}) \star \cdots \star r^(y_n) \star r(z_{j+1}) \star \cdots \star r(z_n)]$. Thus the expression in v contains $r(x_u)$.

If $k+1$ is not the smallest power of $r(x_u)$, then there exists $k' < k$ such that $r^{k'}(x_u)$ is in $E^\star_{t'}(w)$. Applying the recurrence hypothesis, that means that $\mathrm{d}_G(u,w) = k' < k$, which contradicts the hypothesis.

Suppose now that an entity w' has $r^{k+1}(x_u)$ in its expression $E_{t''}(w')$. Then it has an ancestor v' which sent $r^k(x_u)$ to it. Thus, using the recurrence hypothesis, $\mathrm{d}_G(u,v') = k$ and $\mathrm{d}_G(u,w') = k+1$. □

Using the two preceding lemma, we can now show the following theorem. This result should be compared to the one for s-operator (theorem 4).

Theorem 28 *If $\blacktriangledown \in \Delta$ is an idempotent r-operator on A, then the global computation* $\blacktriangledown$**—direct—association** *will stabilize on a deterministic result in whatever network G using whatever data of A, such that the final s-expression on each computing entity v is $E^\star_f(v) = x_v \star r^{d_1}(x_{u_1}) \star \cdots \star r^{d_k}(x_{u_k})$ where x_v is the initial datum of v, u_i, $1 \le i \le k$, are the ancestor entities of v in G, x_{u_i}, $1 \le i \le k$, is the initial datum of u_i, and $d_i = \mathrm{d}_G(u_i, v)$.*

Proof. During the $\blacktriangledown$−**direct−association**, the computing entity v will receive the initial datum x_{u_i} of each its ancestors u_i. Following lemma 27, after a finite time t, $E^\star_t(v)$ will contain $r^{d_i}(x_{u_i})$, where $d_i = \mathrm{d}_G(u_i, v)$.

Now suppose that x_{u_i} is a repeated term in $E^\star_{t'}(v)$ with $t' \ge t$. In that case, following lemma 26, only the smallest power of r is kept. Suppose that this power is $r^d(x_{u_i})$ with $d < d_i$. Then $\mathrm{d}_G(u_i, v) = d < d_i$ (lemma 27), which contradicts the hypothesis.

Hence, the expression in v will no longer change after date t and the global computation stabilizes. □

9 Applications

The idempotent r-operators have nice properties for distributed computations. Indeed, as the s-operators, they allow stabilization after a finite time. In both cases, the time needed to perform a *direct*$-$*association* is at most proportional to the network diameter.

However s-operators give one result per strongly connected components of the network while idempotent r-operators give, for each computing entity, a result depending on its ancestors and the network itself (lemma 27 and theorem 28). Therefore they offer more possibilities for programming, especially to analyze properties of the network.

To illustrate the power of r-operators for distributed computations, we propose the **minc** operator (for *minimum* and *increment*). We denote by $\perp$ the minimum operator on $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$, in its binary form: $x \perp y = \mathbf{min}(x, y)$.

Definition 29 *The operator* **minc**, *denoted in the binary form by* $\blacktriangleleft$, *is defined for all* $x, y \in \overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$ *by:*

$$\begin{aligned} x \blacktriangleleft y &= x \perp (y+1) \qquad \textit{thus} \\ \mathbf{minc}(x, y_1, \ldots, y_n) &= \mathbf{min}(x, y_1+1, \ldots, y_n+1) \end{aligned}$$

Proposition 30 *The* **minc** *operator is an idempotent r-operator on* $\overline{\mathbb{Z}}$ *and verifies the axioms of* Δ.

Proof. The operator $\perp$ is an s-operator on $\overline{\mathbb{Z}}$ and ∞ is its identity element. Moreover, $r : x \mapsto r(x) = x + 1$ is an isomorphism of $(\overline{\mathbb{Z}}, \perp)$: $r(x) \perp r(y) = \mathbf{min}(x+1, y+1) = \mathbf{min}(x, y) + 1 = r(x \perp y)$. Then, following theorem 25, $\blacktriangleleft$ is an r-operator on $\overline{\mathbb{Z}}$ and verifies the axioms of Δ. Finally, $x \blacktriangleleft x = \mathbf{min}(x, x+1) = x$ and $\blacktriangleleft$ is idempotent. $\square$

The **minc** operator can be used for distance computations in networks. For this practical application, we must carefully define the set of the distances, its maximal element denoted by ∞ and the bijective mapping $r : x \mapsto x + 1$. Indeed, it is impossible to work with an infinite set of integers like $\mathbb{Z}$ and to code the maximal element ∞ of $\overline{\mathbb{Z}}$. In fact, it is unnecessary to work with such a set and it is sufficient to take $A = \{0, \ldots, \mathrm{Diam}(G)\}$. The maximal element is then $\infty = \mathrm{Diam}(G)$. Moreover, since we never need $r^{-1}(0)$ and $r(\mathrm{Diam}(G))$, we can define the bijective mapping r from $\{0, \ldots, \mathrm{Diam}(G) - 1\}$ to $\{1, \ldots, \mathrm{Diam}(G)\}$. It is easy to see that the results of this report are still true with these restrictions. The following proposition states that the **minc** operator allows to compute distances.

Proposition 31 *On the network* $G(V, E)$ *initialized with* ∞ *everywhere except in a single source* s *initialized with 0, a* **minc**$-$*direct*$-$*association gives* $\mathrm{d}_G(s, v)$ *in each vertex* v *after stabilization.*

Proof. Following theorem 28, the **minc**$-$*direct*$-$*association* stabilizes and, in each computing entity $v \neq s$ with $k \in \mathbb{N} \setminus \{0\}$ ancestor entities in the network, the result is

$[x_v \perp r^{d_1}(x_{u_1}) \perp \cdots \perp r^{d_k}(x_{u_k})]$ where x_v is the initial data of v, $u_i \in V$ $(1 \leq i \leq k)$ are the ancestors of v, and for all $i \in \{1, \ldots, k\}$, x_{u_i} is the initial datum of u_i and $d_i = \mathrm{d}_G(u_i, v)$. Since s is a source of the digraph $G(V, E)$, there exists $j \in \{1, \ldots, k\}$ such that $u_j = s$. The result in v is then $[\underbrace{\infty \perp \cdots \infty}_{j} \perp r^{\mathrm{d}_G(s,v)} \perp \underbrace{\infty \perp \cdots \perp \infty}_{k-j}] = r^{\mathrm{d}_G(s,v)} = \mathrm{d}_G(s, v)$. In the same way, the result in s is 0. □

This method for distance computation in a network is efficient and can easy be used to build a shortest-path-tree in a network. Algorithms which use distance computation can therefore be improved using the r-operator **minc**.

It is easy to prove the following extension of proposition 31.

Proposition 32 *On the network $G(V, E)$ initialized with ∞ everywhere except in some vertices called* seeds *initialized with 0, a* **minc**$-$*direct*$-$*association gives after stability, for each vertex v, $\mathrm{d}_G(s, v)$ where s is the closest seed of those in ancestors of v, or ∞ if there is no seed in the ancestors of v.*

Proposition 32 allows some applications of r-operators in image analysis. Indeed, the distance transformation is a basic primitive in this field of parallel programming. It is used in skeletization of images by means of median axis transformations or in determination of Voronoï regions *[Tuceryan and Jain, 1990]*.

When such algorithms are programmed on SIMD multi-computers, the underlying topology is often a two-dimensional mesh. In that case, each computing entity manages one pixel of the image. On these architectures, approximated distance transformations are used. For instance the Manhattan distance computes the sum of the X-axis and the Y-axis distances $(|x_1 - x_2| + |y_1 - y_2|)$. Using the **minc** operator, it is easy to compute the Manhattan distance in such a network with a single **minc**$-$*direct*$-$*association*.

When diagonal connections are added to the mesh, approximations of the Euclidean distance can be computed, using a weight of 3 for the diagonal connections and 2 for the others. We can see that such an approximation of the Euclidean distance can be computed in a single global computation.

We denote by $\mathcal{M}$ the mesh with diagonal connection and we number from 1 to 8 the incoming links of the computing entities of $\mathcal{M}$ in such a way that, for each computing entities, the incoming buffers corresponding to diagonal connections get even numbers. To obtain the approximation of the Euclidean distance using a single global computation, we must use the operator $\mathbf{min}(b_0, b_1 + 2, b_2 + 3, b_3 + 2, b_4 + 3, b_5 + 2, b_6 + 3, b_7 + 2, b_8 + 3)$. It cannot be written with a binary operator. However, we can see the global computation stabilizes. Indeed, it is equivalent to perform a **minc**$-$*direct*$-$*association* on a network $\mathcal{M}'$ defined as follows. $\mathcal{M}'$ is obtained from $\mathcal{M}$ by subdividing each horizontal and vertical link with one vertex, and each diagonal link with two vertices.

10 Conclusion

In this report, we have studied stabilization of distributed computations modeled by binary operators. We gave some necessary conditions for $\mathit{direct}-\mathit{association}$ termination. Next we defined r-operators and studied their properties. Finally we showed that r-operators allow $\mathit{direct}-\mathit{association}$ stabilization in any network with any data. As we saw in the last section, r-operators have many applications in distributed computing.

Acknowledgements

I would like to thank Charles Delorme for its interest in my work and for useful suggestions.

References

[Blelloch, 1990] Blelloch, E., G. (1990). *Vector Models for Data-Parallel Computing.* MIT Press, Cambridge, Massachusetts, USA.

[Chang, 1982] Chang, E. J. H. (1982). Echo algorithms: Depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, SE 8(4):391–401.

[Dijkstra and Scholten, 1980] Dijkstra, E. W. and Scholten, C. S. (1980). Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4.

[Ducourthial, 1995] Ducourthial, B. (1995). Etude du modèle réseau associatif. Master's thesis, Laboratoire de Recherche en Informatique, bât. 490, Université Paris-Sud, Orsay, FRANCE. *(Mémoire de DEA Architecture Parallèle).*

[Dulac, 1996] Dulac, D. (1996). *Contribution au Parallélisme Massif en Analyse d'Images : Une Architecture SIMD Fondée sur la Reconfigurabilité et l'Asynchronisme.* PhD thesis, Institut d'Electronique Fondamentale, Université Paris-Sud, 91405 Orsay, FRANCE.

[Hillis, 1985] Hillis, W. D. (1985). *The Connection Machine.* The MIT Press, Cambridge, Massachusetts, USA.

[Hoare et al.,] Hoare, R., Dietz, H., Mattox, T., and Kim, S. Bitwise aggregate networks. `http://garage.ecn.purdue.edu/~papers`.

[Lakshmivarahan and Dhall, 1994] Lakshmivarahan, S. and Dhall, K. S. (1994). *Parallel Computing Using the Prefix Problem.* Oxford University Press.

[Mérigot, 1992] Mérigot, A. (1992). Associative nets: A new parallel computing model. Technical Report 2, Institut d'Electronique Fondamentale, Université Paris Sud, 91405 Orsay, FRANCE.

[Mérigot, 1997] Mérigot, A. (1997). Associative nets: A graph-based parallel computing net. *IEEE Transactions on Computers*, 46(5):558–571.

[Segall, 1983] Segall, A. (1983). Distributed network protocols. *IEEE Transactions on Information Theory*, IT 29(1):23–35.

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

[Tel, 1994] Tel, G. (1994). *Introduction to Distributed Algorithms*, volume 1. Cambridge University Press, Cambridge, UK.

[Tuceryan and Jain, 1990] Tuceryan, M. and Jain, A. (1990). Texture segmentation using Voronoi polygons. *IEEE Transactions on PAMI*, 13:211–216.

Erratum to the report LRI-1140

Proposition 21 is false: $\star$ is not r-commutative. As a consequence, proof of theorem 25 must be replaced by the following one.

Proof. (1) $\Rightarrow$ (2) We only need to prove that r is injective. For all $x, y, a, b \in A$, $r(x) = r(y)$ implies $a \blacktriangledown b \blacktriangledown r(x) = a \blacktriangledown b \blacktriangledown r(y)$. Then $a \blacktriangledown (b \blacktriangledown x) = a \blacktriangledown (b \blacktriangledown y)$ (r-associativity) and thus, using left-cancellation, $x = y$.

(2) $\Rightarrow$ (3) If $\blacktriangledown$ is a binary operator allowing left-cancellation and admitting a right-identity element, then it is in Δ. Thus, $\star$ is an s-operator on A (theorem 20) and r is an isomorphism of $(A, \star)$ (proposition 17).

(3) $\Rightarrow$ (1) If r is an isomorphism of $(A, \star)$, it is a surjective mapping on A, and for all $x, y, z \in A$: $(x \blacktriangledown y) \blacktriangledown r(z) = x \star r(y) \star r^2(z) = x \star (r(y) \blacktriangledown r(z)) = x \blacktriangledown r^{-1}(r(y) \blacktriangledown r(z)) = x \blacktriangledown (y \blacktriangledown z)$ and thus $\blacktriangledown$ is r-associative; $x \blacktriangledown r^{-1}(y) = x \star y = y \star x = y \blacktriangledown r^{-1}(x)$ and $\blacktriangledown$ is r-commutative (proposition 10); $r(x) \blacktriangledown x = r(x) \star r(x) = r(x)$ and thus $\blacktriangledown$ is r-idempotent. Moreover if $x \blacktriangledown y = x \blacktriangledown z$, $x \star r(y) = x \star r(z)$ and thus $y = z$ ($\star$ allows left-cancellation and r is bijective). Thus $\blacktriangledown$ allows left-cancellation on A. Finally, if $e \in A$ is the identity element of $\star$, $x \blacktriangledown r^{-1}(e) = x \star e = x$ and $r^{-1}(e)$ is a right-identity element of $\blacktriangledown$. $\square$