Irreducibility and Additivity of Set Agreement-oriented Failure Detector Classes*

[Extended Abstract] *

Achour Mostefaoui‡ Sergio Rajsbaum§ Michel Raynal‡ Corentin Travers‡
{achour|raynal|ctravers}@irisa.fr rajsbaum@math.unam.mx

ABSTRACT

Solving agreement problems (such as consensus and k-set agreement) in asynchronous distributed systems prone to process failures has been shown to be impossible. To circumvent this impossibility, distributed oracles (also called unreliable failure detectors) have been introduced. A failure detector provides information on failures, and a failure detector class is defined by a set of abstract properties that encapsulate (and hide) synchrony assumptions. Some failure detector classes have been shown to be the weakest to solve some agreement problems (e.g., Ω is the weakest class of failure detectors that allow solving the consensus problem in asynchronous systems where a majority of processes do not crash).

This paper considers several failure detector classes and focuses on their additivity or their irreducibility. It mainly investigates two families of failure detector classes (denoted $\diamond\mathcal{S}_x$ and $\diamond\phi^y$, $0 \leq x,y \leq n$), shows that they can be "added" to provide a failure detector of the class Ω^z (a generalization of Ω). It also characterizes the power of such an "addition", namely, $\diamond\mathcal{S}_x + \diamond\phi^y \rightsquigarrow \Omega^z \Leftrightarrow x+y+z > t+1$, where t is the maximum number of processes that can crash in a run. As an example, the paper shows that, while $\diamond\mathcal{S}_t$ allows solving 2-set agreement (and not consensus) and $\diamond\phi^1$ allows solving t-set agreement (but not $(t-1)$-set agreement), their "addition" allows solving consensus. More generally, the paper studies the failure detector classes $\diamond\mathcal{S}_x$, $\diamond\phi^y$ and Ω^z, and shows which reductions among these classes are possible and which are not. The paper presents also an Ω^k-based k-set agreement protocol. In that sense, it can be seen as a step toward the characterization of the weakest failure detector that allows solving the k-set agreement problem.

*This work has been supported by a grant from LAFMI (Franco-Mexican Lab in Computer Science) and PAPIIT-UNAM.

‡IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France.
§Instituto de Matemáticas, UNAM, D. F. 04510, Mexico.

*A full version of this paper is available in [16].

PODC'06, July 22-26, 2006, Denver, Colorado, USA.

Categories ands Subject Descriptors

C.2.4 [**Computer-Communication Network**]: Distributed Systems -*distributed applications, network operating systems*; D.4.1 [**Operating Systems**] Process Management -*concurrency, multiprocessing, synchronization*; D.4.5 [**Operating Systems**] Reliability -*fault-tolerance*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation -*Computability theory*

General Terms: Algorithms, Reliability, Theory

Keywords: Asynchronous system, Fault-tolerance, Unreliable failure detector.

1. INTRODUCTION

Context of the work: failure detectors for agreement problems. *Consensus* is one of the most fundamental problem in fault-tolerant distributed computing: each process proposes a value, and every non-faulty process must decide a value (termination) such that no two different values are decided (agreement) and the decided value is a proposed value (validity). Despite the simplicity of its definition and its use as a basic building block to solve distributed agreement problems, consensus cannot be solved in asynchronous system where even a single process can crash [7].

Several approaches have been investigated to circumvent this impossibility result. One of them is the failure detector approach [3, 21]. It consists in equipping the underlying system with a distributed oracle that provides each process with (possibly incorrect) hints on process failures. According to the type and the quality of the hints, several classes of failure detectors can be defined. As far as consensus is concerned, two classes are particularly important.

- The class of *leader* failure detectors [2] (denoted Ω). This class includes all the failure detectors that continuously output at each process a process identity such that, after some time, all the correct processes are provided with the same identity that is the identity of a correct process (eventual leadership). Before that time, different processes can be provided with distinct leaders (that can also change over time), and there is no way for the processes to know when this anarchy period is over. Ω-based asynchronous consensus protocols can be found in [8, 12, 19]. (It is important to notice that the first version of the leader-based Paxos protocol dates back to 1989, i.e., before the Ω formalism was introduced.)
- The class of *eventually strong* failure detectors [3] (denoted $\diamond\mathcal{S}$). A failure detector of that class provides each process

with a set of suspected processes such that this set eventually includes all the crashed processes (strong completeness) and there is a correct process p and a time after which no set contains the identity of p (eventual strong accuracy). $\diamond\mathcal{S}$-based asynchronous consensus protocols can be found in [3, 8, 17, 23].

Two important results are associated with these classes. First, they are equivalent (which means that it is possible, from any failure detector of any of these classes, to build a failure detector of the other class) [2, 5, 15]. Second, as far as information on failures is concerned, they are the weakest class of failure detectors that allow solving consensus in asynchronous systems where a majority of processes are correct [2].

The *k-set agreement* problem relaxes the consensus requirement to allow up to k different values to be decided [4] (consensus is 1-set agreement). This problem is solvable in asynchronous system despite up to $k-1$ process crash failures, but has been shown to be impossible to solve as soon as k or more processes can crash [1, 11, 22].

The failure detector class $\diamond\mathcal{S}$ has been weakened in [18, 24] to address this problem. While the scope of the accuracy property of $\diamond\mathcal{S}$ spans the whole system (there is a correct process that, after some time, is not suspected by any process), the class $\diamond\mathcal{S}_x$ is defined by the same completeness property and a limited scope accuracy property, namely, there is a correct process that, after some time, is not suspected by x processes. It is easy to see that $\diamond\mathcal{S}_n$ (where n is the total number of processes) is $\diamond\mathcal{S}$, while $\diamond\mathcal{S}_1$ provides no information on failures. Moreover, $\diamond\mathcal{S}_{x+1} \subseteq \diamond\mathcal{S}_x$. It has been shown that, when we consider the family $(\diamond\mathcal{S}_x)_{1\leq x\leq n}$ of failure detectors, $\diamond\mathcal{S}_x$ is the weakest class that allows solving k-set agreement in asynchronous systems for $k = t-x+2$ (where t is an upper bound on the number of processes that can crash) [10] (message-passing systems must also satisfy the additional constraint of a majority of correct processes, $t < n/2$). The class $\mathcal{S}_x$ is a subset of $\diamond\mathcal{S}_x$. It has a the same completeness property but a stronger accuracy property: it requires that, from the very beginning, there is a subset of x processes that never suspect one correct process.

A new family of failure detectors (denoted $(\phi^y)_{0\leq y\leq n}$), has recently been introduced in [14] (where it is used in conjunction with conditions [13] to solve set agreement problems). A failure detector of the class ϕ^y provides the processes with a query primitive that has a parameter (a set X of processes) and returns a boolean answer. The invocation QUERY(X) by a process returns systematically *true* (resp., *false*) when $0 \leq |X| \leq t-y$, i.e., when the set is too small (resp., $|X| > t$, i.e., when the set is too big). When $t - y < |X| \leq t$ (the set has then an appropriate size), if QUERY(X) returns *true* then all the processes in X have crashed; moreover, if all the processes of X have crashed and a process repeatedly issues QUERY(X), it eventually obtains the answer *true*. We have $\phi^{y+1} \subseteq \phi^y$. Moreover, ϕ^0 provides no information on failures, while, $\forall\ y \geq t$, ϕ^y is equivalent to a perfect failure detector (one that never makes a mistake [3]). It is shown in [14], that, in shared memory systems, ϕ^y is the weakest failure detector class of the family $(\phi^y)_{0\leq y\leq t}$ that allows solving asynchronous k-set agreement with $k = t - y + 1$.

The family of failure detector classes $(\Omega^z)_{1\leq z\leq n}$ [20] has been introduced to augment the synchronization power of object types in the wait-free hierarchy. A failure detector of the class Ω^z outputs at each process a set of at most z process identities such that, after some time, the same set including the identity of at least one correct process is output at all correct processes. Clearly, Ω^1 is Ω. Moreover, $\Omega^z \subseteq \Omega^{z+1}$.

Motivation and results. The paper first extends the family $(\phi^y)_{0\leq y\leq n}$, by considering its eventual counterpart, namely the family $(\diamond\phi^y)_{0\leq y\leq n}$. $\diamond\phi^y$ is a weakening of ϕ^y in the sense it allows the properties defining ϕ^y to be satisfied only after some finite time. So, while the families $(\mathcal{S}_x)_{1\leq y\leq n}$ and $(\phi^y)_{0\leq y\leq n}$ are characterized by a "perpetual" property (i.e., a property that has to be satisfied from the very beginning), the families $(\diamond\mathcal{S}_x)_{1\leq y\leq n}$, $(\Omega^z)_{1\leq z\leq n}$ and $(\diamond\phi^y)_{0\leq y\leq n}$ are characterized by an eventual property.

It appears that, when we are interested in solving set agreement problems, we are provided with three families of failure detectors: $(\diamond\mathcal{S}_x)_{1\leq x\leq n}$, $(\diamond\phi^y)_{0\leq y<n}$ and $(\Omega^z)_{1\leq z\leq n}$. Whatever the problems these failure detector classes help solving, important questions are the following: *Which among these classes are equivalent? Which are not? Is it possible to combine some of them to obtain stronger failure detector classes? If the answer is "yes", which ones and which failure detector class do they produce? If the answer is "no", why? Etc.* This is the type of questions addressed in this paper that characterizes relationships linking each pair of failure detector classes. More precisely, the contributions of the paper are the following. The notation $A + B \rightsquigarrow C$ means that, given as inputs a failure detector of the class A and a failure detector of the class B, there is an algorithm that constructs a failure detector of the class C. The notation $A + B \not\rightsquigarrow C$ means that there is no such transformation algorithm. The notations $A \rightsquigarrow C$ and $A \not\rightsquigarrow C$ have the same meaning considering a single failure detector class as input.

CONTRIBUTION C1: Reducibility, Irreducibility and Minimality.

- Relations linking $\phi^y/\diamond\phi^y$ and $\mathcal{S}_x/\diamond\mathcal{S}_x$:
  - Let $1 \leq x \leq t+1$ and $1 \leq y \leq t$. $\mathcal{S}_x \not\rightsquigarrow \diamond\phi^y$. (Theorem 5.)
  - Let $0 \leq y < t$ and $1 < x \leq t+1$. $\phi^y \not\rightsquigarrow \diamond\mathcal{S}_x$. (Theorem 6.)
- Relations linking $\phi^y/\diamond\phi^y$ and Ω^z:
  - $\diamond\phi^y \rightsquigarrow \Omega^z$ Iff $y + z > t$. (Corollary 4.)
  - Let $1 \leq z \leq t+1$ and $1 \leq y \leq t$. $\Omega^z \not\rightsquigarrow \diamond\phi^y$. (Theorem 7.)
- Relations linking $\diamond\mathcal{S}_x$ and Ω^z:
  - $\diamond\mathcal{S}_x \rightsquigarrow \Omega^z$ Iff $x + z > t + 1$. (Corollary 5.)
  - Let $1 < x, z \leq t$. $\forall z : \Omega^z \not\rightsquigarrow \diamond\mathcal{S}_x$. (Theorem 8.)

All these relations are depicted in Figure 1 where the bold arrows mean reducibility, and the dotted arrows mean irreducibility. The class $\mathcal{S}_x$ is the subclass of $\diamond\mathcal{S}_x$ where the accuracy is perpetual (namely, there is a correct process that is not suspected by x processes from the very beginning). $\mathcal{P}$ is the class of *perfect* failure detectors [3] (the ones that never make a mistake). The column at the right of the figure concerns k-set agreement: all the failure detector classes in the zth line allow solving z-set agreement. It is important to notice that (1) $\diamond\mathcal{S}_{t-z+2}$ and $\diamond\phi^{t-z+1}$ cannot be compared, (2) both are stronger than Ω^z. Moreover, given a line (say z) of the figure, Ω^z is the weakest class of that line that allows solving k-set agreement.

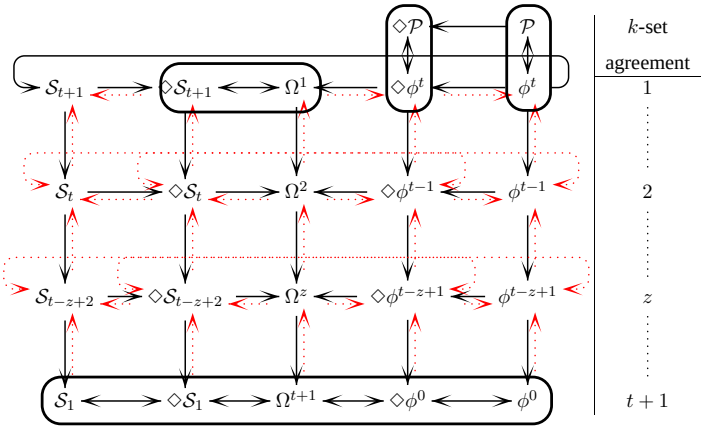

Figure 1: Grid of failure detector classes

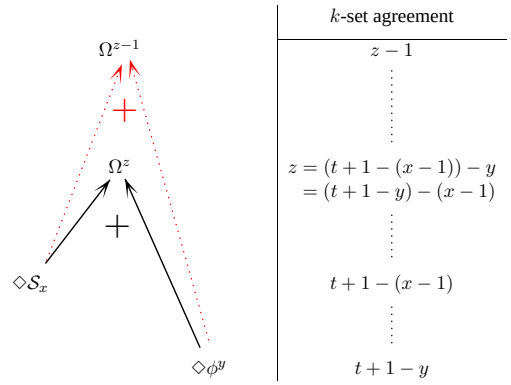

Figure 2: Additivity of $\diamond\mathcal{S}_x$ and $\diamond\phi^y$

CONTRIBUTION C2: Additivity. This paper poses the question of adding failure detectors of distinct classes. This is an important issue as "additivity" is a crucial concept as soon as modularity and scalability of distributed systems are concerned.

As an example, assuming $t > 1$, let us consider the class $\diamond\mathcal{S}_t$ that allows solving 2-set agreement (but not consensus), and the class $\diamond\phi^1$ that allows solving t-set agreement (but not $(t-1)$-set agreement). What about $\diamond\mathcal{S}_t + \diamond\phi^1$? Is it possible to add them? If the answer is "yes", which type of information on failures is provided by their combination? The paper shows that $\diamond\mathcal{S}_t + \diamond\phi^{t-1}$ allows solving the consensus problem. More generally, with respect to the grid described in the previous figure, the paper characterizes which classes can be added and which cannot. More explicitly, it shows the following result (see also Figure 2): $\boxed{\diamond\mathcal{S}_x + \diamond\phi^y \rightsquigarrow \Omega^z \Leftrightarrow x+y+z > t+1.}$ To that end, the paper presents a construction algorithm (sufficiency part, Figures 4 and 5), and an impossibility proof (necessity part, Theorem 4).

Intuitively, this shows that $\diamond\mathcal{S}_x$ and $\diamond\phi^y$ provide different seeds to build Ω^z. To see the gain provided by such an addition, let us analyze it as follows:
- As $\diamond\mathcal{S}_x \rightsquigarrow \Omega^{t-x+2}$, the previous addition shows that adding $\diamond\phi^y$ allows strengthening Ω^{t-x+2} to obtain Ω^z with $z = (t-x+2)-y$.
- Similarly, as $\diamond\phi^y \rightsquigarrow \Omega^{t-y+1}$, the previous addition shows that adding $\diamond\mathcal{S}_x$ allows strengthening Ω^{t-y+1} to Ω^z with $z = (t-y+1)-(x-1)$.

CONTRIBUTION C3: Asynchronous Ω^k-based k-set agreement. This paper proposes such an algorithm. (To our knowledge, no previous work has addressed the design of Ω^z-based k-set agreement algorithms.) The proposed algorithm is very simple. Moreover, the paper establishes that $t < n/2$ and $z \leq k$ are two tight bounds of the k-set agreement problem, when considering the $(\Omega^z)_{1\leq z\leq n}$ family of failure detectors in an asynchronous message-passing system (Theorem 1). Consequently, among all the classes described in Figure 1, Ω^k is the weakest class for solving asynchronous k-set agreement (hence, the algorithm is optimal in that respect). This constitutes a step towards the characterization of the weakest failure detector class that allows solving the k-set agreement problem. Due to space limitation, the protocol is given in [16].

From a methodology point of view, the paper uses as much as possible reduction algorithms (striving not to reinvent the wheel). Two more transformations for particular cases are presented in [16]. These transformations are simpler and more efficient than the general transformation building a failure detector of the class Ω^z from failure detectors of the classes $\diamond\mathcal{S}_x$ and $\diamond\phi^y$. The first transforms ϕ^y into Ω^z for $y+z > t$. The second presents an addition algorithm of $\diamond\mathcal{S}_x$ with $\diamond\phi^y$ that provides $\diamond\mathcal{S}$ when $x+y > t$.

Roadmap. The paper is made up of 5 sections. Section 2 describes the asynchronous computing model and the classes of failure detectors we are interested in. Section 3 presents lower bound on the solvability of the k-set agreement problem in asynchronous systems equipped with a failure detector of the family $(\Omega^z)_{1\leq z\leq n}$. Then, Section 4 presents an algorithm that builds a failure detector of the class Ω^z from a pair of underlying failure detectors, one of the class $\diamond\phi^y$, the other of the class $\diamond\mathcal{S}_x$. Section 5 shows that $x+y+z > t+1$ is a necessary requirement for the previous construction, and establishes the irreducibility relations depicted by the grid of Figure 1. Due to page limitations, (1) the proof of the transformation algorithm and (2) the k-set agreement protocol based on Ω^k are given in [16].

2. COMPUTATION MODEL

2.1 Asynchronous System with Process Crash Failures

We consider a system consisting of a finite set Π of $n \geq 2$ processes, namely, $\Pi = \{p_1, p_2, \ldots, p_n\}$. When it is not ambiguous we also use Π to denote the set of the identities $1, \ldots, n$ of the processes. A process can fail by *crashing*, i.e., by prematurely halting. It behaves correctly (i.e., according to its specification) until it (possibly) crashes. By definition, a process is *correct* in a run if it does not crash in that run; otherwise it is faulty. As previously indicated, t denotes the maximum number of processes that can crash in a run ($1 \leq t < n$). The identity of the process p_i is i, and each process knows all the identities.

Processes communicate and synchronize by sending and receiving messages through channels. Every pair of processes is connected by a channel. Channels are assumed to be reliable: they do not create, alter or lose messages. In particular, if p_i sends a message to p_j, then eventually p_j receives that message unless it fails. There is no assumption about the relative speed of processes or message transfer delays (let us observe that channels are not required to be FIFO).

Broadcast(m) is an abbreviation for "**foreach** $p_j \in \Pi$ **do** *send*(m) *to* p_j **enddo**". Moreover, we assume (without loss of generality) that the communication system provides the processes with a *reliable broadcast* abstraction [9]. Such an abstraction is made up of two primitives *Broadcast()* and *Deliver()* that allow a process to broadcast and deliver messages (we say accordingly that a message is R_broadcast or R_delivered by a process) and satisfy the following properties:

- Validity: If a process R_delivers m, then some process has R_broadcast m. (No spurious messages.)
- Integrity: A process R_delivers a message m at most once. (No duplication.)
- Termination: If a message m is R_broadcast or R_delivered by a correct process, then all the correct processes R_delivers m. (No message R_broadcast or R_delivered by a correct process is missed by a correct process.)

As we can see, the messages sent (resp., R_broadcast) by a process are not necessarily received (resp., R_delivered) in their sending order. Moreover, different processes can R_deliver messages in different order. There is no assumption on message transfer delays. The communication system is consequently reliable and asynchronous.

2.2 Failure Detector Classes

The definition of the families of failure detector classes $(\mathcal{S}_x)_{1\leq x\leq n}$, $(\diamond\mathcal{S}_x)_{1\leq x\leq n}$, $(\phi^y)_{0\leq y<n}$ and $(\Omega^z)_{1\leq z\leq n}$ have been sketched in the introduction. This section provides more complete definitions.

The classes $(\mathcal{S}_x)_{1\leq x\leq n}$ ***and*** $(\diamond\mathcal{S}_x)_{1\leq x\leq n}$**.** A failure detector of the class $\mathcal{S}_x$ or $\diamond\mathcal{S}_x$ consists of a set of modules, each one attached to a process: the module attached to p_i maintains a set (named $suspected_i$) of processes it currently suspects to have crashed. As in other papers devoted to failure detectors, we say "process p_i suspects process p_j at some time τ", if $p_j \in suspected_i$ at that time. Moreover, (by definition) a crashed process suspects no process.

The failure detector $\diamond\mathcal{S}_x$ [18, 24] class generalizes the class $\diamond\mathcal{S}$ defined in [3] (we have $\diamond\mathcal{S}_n = \diamond\mathcal{S}$). A failure detector belongs to the class $\diamond\mathcal{S}_x$ if it satisfies the following properties:

- Strong Completeness: Eventually, every process that crashes is permanently suspected by every correct process.
- Limited Scope Eventual Weak Accuracy: There is a time after which there is a set Q of x processes such that Q contains a correct process and that process is never suspected by the processes of Q.

Similarly, the class $\mathcal{S}_x$ generalizes the class $\mathcal{S}$ [3] (and we have $\mathcal{S}_n = \mathcal{S}$). A failure detector of the class $\mathcal{S}_x$ satisfies the previous strong completeness property, plus the following accuracy property:

- Limited Scope Perpetual Weak Accuracy: there is a set Q of x processes such that (from the very beginning) Q contains a correct process and that process is never suspected by the processes of Q.

It is easy to see that $\mathcal{S}_{x+1} \subseteq \mathcal{S}_x$, $\diamond\mathcal{S}_{x+1} \subseteq \diamond\mathcal{S}_x$, and $\mathcal{S}_x \subseteq \diamond\mathcal{S}_x$. It is also easy to see that the failure detectors of the classes $\mathcal{S}_1$ and $\diamond\mathcal{S}_1$ provide no information on failures. It has been shown in [10] that $\diamond\mathcal{S}_x$ is the weakest failure detector class of the family $(\diamond\mathcal{S}_x)_{1\leq x\leq n}$ that allows solving k-set agreement for $k = t - x + 2$, in asynchronous message-passing systems with a majority of correct processes ($t < n/2$).

The classes $(\Omega^z)_{1\leq z\leq n}$**.** This family of failure detectors has been introduced in [20]. A failure detector of the class Ω^z maintains at each process p_i a set of processes of size at most z (denoted $trusted_i$) that satisfies the following property:

- Eventual Multiple Leadership: there is a time after which the sets $trusted_i$ of the correct processes contain forever the same set of processes and at least one process of this set is correct.

The family $(\Omega^z)_{1\leq z\leq n}$ generalizes the class of failure detectors Ω defined in [2] (we have $\Omega^1 = \Omega$).

Recently, another generalization of Ω has been studied in [6] that considers Ω_S, where S is a predefined subset of the processes of the system. Ω_S requires that all the correct processes of S eventually agree on the same correct leader (it is not required that their eventual common leader belongs to S). Let X be the set of all the pairs of processes. It is shown in [6] that, given all the Ω_x, $x \in X$, it is possible to build Ω.

The classes $(\phi^y)_{0\leq y<n}$**.** These failure detector classes have been introduced in [14] to solve set agreement problems in combination with conditions [13]. Differently from the previous classes of failure detectors that provides each process p_i with a set ($suspected_i$ or $trusted_i$) that p_i can only read, a failure detector of a class ϕ^y provides the processes with a primitive QUERY(X), where X is a set of process identities supplied by the invoking process. Such a primitive allows a process p_i to query about the crash of a region X of the system. More precisely, a failure detector of the class ϕ^y is defined by the following properties (remind that t is an upper bound on the number of process crashes):

- Triviality property. If $|X| \leq t - y$, then QUERY$_y(X)$ returns *true*. If $|X| > t$, then QUERY(X) returns *false*.
- Safety property. If $t - y < |X| \leq t$, then if at least one process in X has not crashed when QUERY(X) is invoked, the invocation returns *false*.
- Liveness property. Let X be such that $t - y < |X| \leq t$. Let τ be a time such that, at time τ, all the processes in X have crashed. Moreover, let us assume that after τ there is an infinite sequence of invocations of QUERY(X). Then, for some time $\tau' \geq \tau$, all the invocations of QUERY(X) return *true*.

The triviality property provides the invoking process with a trivial output when the set X is too small or too big. The safety property states that if the output is *true*, then all the processes in X have crashed. The liveness property states that QUERY(X) eventually outputs *true* when all the processes in X have crashed. It is shown in [14] that (1) $\phi^{y+1} \subseteq \phi^y$, and (2) ϕ^t and the class $\mathcal{P}$ of perfect failure detectors are equivalent in any system where at most t processes can crash. Moreover, it is easy to see that ϕ^0 provides no information on failures. Within the family $(\phi^y)_{0\leq y\leq t}$ of failure detector classes, ϕ^y is the weakest for solving k-set agreement for $k = t-y+1$, in asynchronous shared memory systems.

The classes $(\diamond\phi^y)_{0\leq y<n}$. The failure detector class $\diamond\phi^y$ is the "eventual" counterpart of the class ϕ^y. More precisely, a failure detector of the class $\diamond\phi^y$ is defined by the following properties (remind that t is an upper bound on the number of process crashes):

- Triviality property. If $|X| \leq t - y$, then QUERY$_y(X)$ returns *true*. If $|X| > t$, then QUERY(X) returns *false*.
- Eventual Safety property. Let X be such that $t - y < |X| \leq t$. Suppose that at least one correct process belongs to X. Moreover, let us assume that there is an infinite sequence of invocation of QUERY(X). Then, it exists some time τ from which all the invocations of QUERY(X) return *false*.
- Liveness property. Let X be such that $t - y < |X| \leq t$. Let τ be a time such that, at time τ, all the processes in X have crashed. Moreover, let us assume that after τ there is an infinite sequence of invocations of QUERY(X). Then, for some time $\tau' \geq \tau$, all the invocations of QUERY(X) return *true*.

As for the classes $(\phi^y)_{0\leq y\leq t}$, it follows from these properties that (1) $\diamond\phi^{y+1} \subseteq \diamond\phi^y$, and (2) $\diamond\phi^t$ and the class $\diamond\mathcal{P}$ are equivalent in any system where at most t processes can crash.

Notation. Let $\mathcal{F}$ and $\mathcal{G}$ be any two classes among the previous classes of failure detectors. The notation $\mathcal{AS}_{n,t}[\mathcal{F}]$ is used to represent a message-passing asynchronous system made up of n processes, where up to t may crash ($1 \leq t \leq n$), equipped with a failure detector of the class $\mathcal{F}$. Similarly, $\mathcal{AS}_{n,t}[\mathcal{F}, \mathcal{G}]$ denotes a system equipped with a failure detector of the class $\mathcal{F}$ and a failure detector of the class $\mathcal{G}$. Finally, $\mathcal{AS}_{n,t}[\emptyset]$ denotes a "pure" asynchronous message-passing system (i.e., without additional equipment).

3. FROM Ω^K TO K-SET AGREEMENT

3.1 A k-Set Agreement Algorithm

As announced in the introduction, it is possible to design an Ω^k-based k-set agreement algorithm. Due space limitation, such an algorithm (inspired from an Ω-based consensus protocol [8]) with its proof are described in [16]. This algorithm assumes $t < n/2$.

3.2 A Lower Bound

Considering an asynchronous message-passing system equipped with a failure detector of the class Ω^z, $1 \leq z \leq n$, this section establishes that $t < n/2$ and $z \leq k$ are necessary and sufficient conditions for solving the k-set agreement problem. As already noticed, this result is obtained by a reduction to the problem of the weakest failure detector in the family $(\diamond\mathcal{S}_x)_{1\leq x\leq n}$ that allows solving k-set agreement.

THEOREM 1. *The k-set agreement problem is solvable in $\mathcal{AS}_{n,t}[\Omega^z]$ if and only if $t < n/2$ and $z \leq k$.*

Proof [$\Rightarrow$ part] The proof is by contradiction. let us assume that there is an algorithm $\mathcal{A}$ that solves the k-set agreement problem in $\mathcal{AS}_{n,t}[\Omega^z]$ such that $t \geq n/2$ or $z > k$. Due to Theorem 5, there is an algorithm $\mathcal{T}$ that builds a failure detector of the class Ω^z in $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_{t-z+2}]$. Moreover, there are such transformation algorithms (e.g., the one presented in Section 4 with $y = 0$) that are independent of the value of t (i.e., $t < n$). Combining such a transformation $\mathcal{T}$ and the algorithm $\mathcal{A}$, we obtain an algorithm that solves the k-set agreement problem in $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_{t-z+2}]$. It then follows from the lower bound established by Herlihy and Penso [10] for solving the k-set agreement problem in $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_{t-z+2}]$ that $t < \min(n/2, (t - z + 2) + k - 1)$, from which we conclude $t < n/2$ and $z \leq k$: a contradiction.
[$\Leftarrow$ part] This part follows directly from the very existence of the Ω^k-based k-set agreement algorithm described in [16].
$\square_{Theorem\ 1}$

4. ADDITIVITY OF THE FAILURE DETECTOR CLASSES $\diamond\mathcal{S}_X$ AND $\diamond\phi^Y$

This section presents an algorithm that, given as input any pair of failure detectors of the classes $\diamond\mathcal{S}_x$ and $\diamond\phi^y$, constructs a failure detector of the class Ω^z, provided that $x + y + z > t + 1$. (It is proved in Section 5.1 that this is a necessary requirement for such a construction, thereby showing that the algorithm is optimal.)

The algorithm is made up of two components that we call *wheels* because each "turns" like a gear-wheel until they become synchronized and stop turning. The wheel that is the first to eventually stop is the one whose progress depends on the the underlying $\diamond\mathcal{S}_x$ failure detector ("lower" wheel). When it stops, it provides a property that allows the second wheel in turn to eventually stop ("upper" wheel). As we will see, the wheel metaphor comes from the fact that each component is made up of main tasks that "turn", each scanning a sequence until some property becomes satisfied.

Let us remind that $1 \leq x \leq n$. Moreover, as the class $\diamond\phi^t$ is equivalent to the class of eventual perfect failure detectors we consider only the cases $0 \leq y \leq t$, from which we conclude $t - y + 1 > 0$. Finally, as $z \geq t + 2 - (x + y)$ is a necessary requirement and Ω^1 is the strongest class in the family $(\Omega^z)_{1\leq z\leq n}$, the only interesting cases for the pair (x, y) are when $t + 2 - (x + y) \geq 1$. Hence, in the following we consider that $t - y + 1 > 0$, $z = t + 2 - (x + y)$ and $t + 2 - (x + y) > 0$.

4.1 The Lower Wheel Component

The aim of this component is to provide each process p_i with a local variable $repr_i$ intended to contain a process identity and such that the following property becomes eventually satisfied: there is a set X of x processes that either have crashed, or the variables $repr_i$ of the processes of X that have not crashed, contains the identity ℓx of one of

them that is a correct process. This process is their common representative (leader). The variable $repr_i$ of a process p_i that does not belong to X has to be equal to the identity i of p_i.

To attain this goal the processes use their sets $suspected_i$ that collectively satisfy the completeness and limited scope eventual accuracy properties defining the class $\diamond\mathcal{S}_x$. Let $\mathcal{X}$ be the finite set of all the sets of x processes that can be built from the set $\Pi = \{p1, \ldots, p_n\}$. Let nb_x denote the number of combinations of x elements in a set of n elements. $\mathcal{X}$ has nb_x elements. Let us organize $\mathcal{X}$ as a sequence, and let $\mathcal{X}[k]$ be its kth element, $1 \leq k \leq nb\_x$. Within $\mathcal{X}[k]$, let us arrange the x processes it is made up of in some predefined (arbitrary) order: $\ell_1^k, \ldots, \ell_x^k$. This means that the infinite sequence $\mathcal{X}[1], \mathcal{X}[2], \ldots, \mathcal{X}[nb\_x], \mathcal{X}[1], \ldots$ gives rise to an infinite sequence of process identities, namely, $\ell_1^1, \ldots, \ell_x^1, \ell_1^2, \ldots, \ell_x^2, \ell_1^3, \ldots$ (see Figure 3). This sequence is assumed to be initially known by all the processes in order they can scan it in the same order.

In addition to its output $repr_i$, each process p_i manages a local set X_i and a local variable ℓx_i. It starts with X_i initialized to $\mathcal{X}[1]$, and ℓx_i initialized to ℓ_1^1 (the first process of $\mathcal{X}[1]$). Then, it uses the function $\mathsf{Next}(-,-)$ defined as follows to progress along the infinite sequence of process identities. $\mathsf{Next}(\ell_y^k, \mathcal{X}[k])$ outputs the pair $(\ell_{y+1}^k, \mathcal{X}[k])$ if $y < x$ and the pair $(\ell_1^{k+1}, \mathcal{X}[k+1])$ if $y = x$ (with $k+1$ being replaced by 1 when $k = nb\_x$).

The behavior of the lower wheel component of a process p_i is described in Figure 4. It is made up of two simple tasks. The processes scan the infinite sequence of sets generated from $\mathcal{X}$ until they stabilize. X_i represents the set of x processes that are currently in charge of extracting a common representative ℓx_i from this set. To do it, each process p_i that belongs to X_i uses its set $suspected_i$ provided by the underlying failure detector of the class $\diamond\mathcal{S}_x$. If the processes of X_i succeed in not suspecting one of them -whose identity is kept by p_i in ℓx_i-, they stop sending X_MOVE() messages. Differently, if a process p_j of the set X_i suspects its current "leader" ℓx_j, it uses the reliable broadcast primitive to send the message X_MOVE$(\ell x_j, X_i)$ indicating that, from its point of view, ℓx_j cannot be their common representative. A process p_j delivers a message X_MOVE$(\ell x, X)$ only when $\ell x = \ell x_i$ and $X_i = X$; it then proceeds to the next process identity (according to the infinite sequence), and possibly to the next candidate set $\mathcal{X}[k+1]$ if $X_i = X = \mathcal{X}[k]$ and $\ell x = \ell x_i$ is the last process of $\mathcal{X}[k]$.

Let us finally consider that the processes progress until they consider a set X such that the x processes that constitute X have crashed. It is easy to see that each non-crashed process p_i continues looping inside task $T1$ without sending messages, and is such that $repr_i = i$.

Proof of the lower wheel component. The proof considers an arbitrary run of the algorithm described in Figure 4. C denotes the set of processes that are correct in that run. Moreover, var_i^τ denotes the value of the local variable var_i at time τ.

LEMMA 1. *$\forall i \in C$, there are a pair (λ_i, σ_i) and a time τ_i such that $\forall \tau \geq \tau_i : (\ell x_i^\tau, X_i^\tau) = (\lambda_i, \sigma_i)$.*

COROLLARY 1. *The protocol is quiescent (i.e., eventually all the processes stop sending* X_MOVE *messages).*

LEMMA 2. *$\forall i, j \in C : (\lambda_i, \sigma_i) = (\lambda_j, \sigma_j)$. (In the following, (λ, σ) denotes that pair.)*

LEMMA 3. *$(\sigma \cap C \neq \emptyset) \Rightarrow (\lambda \in C)$.*

THEOREM 2. *The algorithm described in Figure 4 ensures the existence of a set X and a time τ such that $\forall \tau' \geq \tau$, the following holds:*
1. $|X| = x$,
2. $i \in \Pi - X \Rightarrow repr_i = i$,
3. $\forall i, j \in X \cap C : repr_i = repr_j = \rho \in C \cap X$.

Proof Let $\tau = \max\{\tau_i : i \in \Pi\}$ where τ_i is the time introduced in Lemma 1, and σ and λ be the set and the process identity defined in Lemma 2. Let us first observe that due to its definition (σ is a set X_i) we have $|\sigma| = x$ (1). Let p_i be a correct process. If $i \in \Pi - X$, then as the value of $repr_i$ does not change after time τ (Lemma 1 and Task $T1$), it follows that $repr_i = i$ is permanently true from time τ (2). Moreover, it directly follows from Lemma 2 and task $T1$ that all the correct processes p_j belonging to the set σ have permanently the same representative $repr_j = \lambda$ from time τ. Finally, λ is the identity of a correct process due to Lemma 3 (3). Taking $X = \sigma$, $\tau = \max\{\tau_i : i \in \Pi\}$ and $\rho = \lambda$ completes the proof of the theorem. $\square_{Theorem\ 2}$

4.2 The Upper Wheel Component

The "upper wheel" component consists of four tasks $T3 - T6$ (Figure 5). Similarly to the lower wheel component, it uses a set, that we call $\mathcal{Y}$, including all the possible sets of $t-y+1$ processes built from the n processes composing the system. Let nb_y denote the number of such distinct sets. Organizing $\mathcal{Y}$ as a sequence, let $\mathcal{Y}[k]$ be its kth element, and let us consider the infinite sequence $\mathcal{Y}[1]$,$\mathcal{Y}[2], \ldots, \mathcal{Y}[nb\_y]$, $\mathcal{Y}[1], \ldots$ Moreover, given any set $\mathcal{Y}[k]$ of this sequence, let us consider all its subsets of size $z = (t+2) - (x+y)$ (let nb_L denote the number of such subsets). Finally, let us order them (the order is arbitrary). Let $L_1^k, L_2^k, \ldots, L_{nb\_L}^k$ denote the sequence of all the sets of size $(t+2)-(x+y)$ generated from the set $\mathcal{Y}[k]$ (whose size is $t-y+1$). As before, the infinite sequence $L_1^1, L_2^1, \ldots, L_{nb\_L}^1, L_1^2, \ldots, L_{nb\_L}^2, \ldots, L_1^{nb\_y}, \ldots, L_{nb\_L}^{nb\_y}, L_1^1, L_2^1, \ldots$ is initially known by each process. The function $\mathsf{Next}(L_r^k, \mathcal{Y}[k])$ is defined similarly to the previous $\mathsf{Next}()$ function. It outputs $(L_{r+1}^k, \mathcal{Y}[k])$ if $r < nb\_L$, and outputs $(L_1^{k+1}, \mathcal{Y}[k+1])$ if $r = nb\_L$ ($k+1$ being replaced by 1 when $k = nb\_y$).

Given these ingredients we can now describe the principles the additive transformation relies on. The aim is for p_i to return L_i as the value of the set $trusted_i$ it provides to the upper layer (remind that this set has to include at most z processes and eventually at least one correct process). Within the upper wheel component, the processes start from the set $\mathcal{Y}[1]$ and then scan the same infinite sequence of sets $\mathcal{Y}[1], \mathcal{Y}[2], \ldots, \mathcal{Y}[nb\_y], \mathcal{Y}[1], \ldots$ (tasks $T3$ and $T4$). When p_i is working with Y_i, it looks for one of its subset L_i (of size z) containing a correct process (when this occurs, that set defines the value of $trusted_i$). To check if its current set L_i contains a correct process, p_i sends INQUIRY() messages and waits until it has received at least one RESPONSE(id) message from a process of Y_i or all the processes of the set Y_i have crashed (task $T3$). When a process p_j sends back a

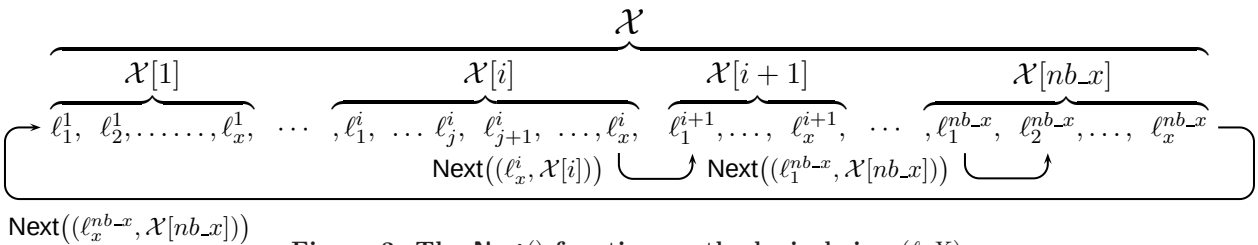

Figure 3: The Next() function on the logical ring (ℓ, X)

Init: $X_i \leftarrow \mathcal{X}[1]$; $\ell x_i \leftarrow \ell_1^1$; $repr_i \leftarrow i$
Task $T1$:
  repeat forever
    if $(i \in X_i)$ **then** $repr_i \leftarrow \ell x_i$ **else** $repr_i \leftarrow i$ **end_if**;
    if $\big((i \in X_i) \wedge (\ell x_i \in suspected_i)\big)$ **then** *R_Broadcast* X_MOVE$(\ell x_i, X_i)$ **end_if**
  end_repeat
Task $T2$: **when** X_MOVE$(\ell x_i, X_i)$ **is R_delivered**: $(\ell x_i, X_i) \leftarrow \mathsf{Next}(\ell x_i, X_i)$

Figure 4: From $\diamond\phi^y + \diamond\mathcal{S}_x$ to Ω^z: lower wheel component (code for p_i)

response, it sends the last identity $repr_j$ currently computed by its underlying wheel (task $T5$). Let us consider two cases.

- Case A. The first case is when all the processes of Y_i have crashed (ϕ-QUERY(Y_i) then eventually returns *true*, lines 04 and 09). It follows that the task $T3$ stops broadcasting *R_Broadcast* L_MOVE(L_i, Y_i) messages. In that case, the value returned for $trusted_i$ (line 09) is the smallest identity among the non-crashed processes.
- Case B. The second case is when p_i receives a response message from a process in Y_i. Then, the set rec_from_i is not empty and contains the identities of the representative $repr_j$ of each process p_j that has answered. We consider two subcases.
  - Case B.1. None of these identities belongs to L_i. p_i then suspects that all the processes of L_i have crashed. It consequently broadcasts the message *R_Broadcast* L_MOVE(L_i, Y_i) to entail the progress of all the processes to the next L_i set.
  - Case B.2. One of these identities belongs to L_i. In that case p_i considers that its current set L_i contains one correct process (the one with that identity). It then continues sending INQUIRY() messages until either none of the identities it receives belongs to L_i (and then we are in case B.1), or it receives a *R_Broadcast* L_MOVE(L_i, Y_i) message which entails its progress to the next L_i.

Let us notice that, due to the property eventually ensured on the $repr_j$ local variables by the lower wheel component, there is a time after which all the RESPONSE(id) messages carry identities of correct processes. It follows that if the set L_i currently investigated by the processes does not change, that set includes at least one correct process and we have obtained the property required for $trusted_i$.

To capture the intuition that underlies the fact that the two wheels synchronize and the processes stabilize on the same set L, let us first recall that, due to the properties of the lower wheel component, there is a time after which there is a set X of x processes such that (1) either all its processes have crashed, or (2) each non-crashed process p_j of X is such that $repr_j = \ell x$ (the identity of a correct process of X). In both cases, a process p_i that does not belong to X is then such that $repr_i = i$.

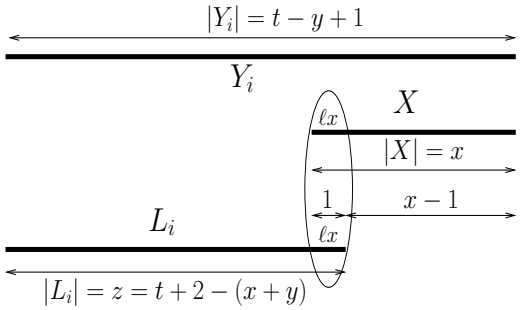

Figure 6: When the upper wheel stops looking for a new L_i set

Let us examine the configuration described in Figure 6. We show that in this configuration a process p_i cannot entail the progress from (L_i, Y_i) to $\mathsf{Next}(L_i, Y_i)$. As this is true for any process p_i, it follows that the processes converge to the same final leader set L_i. The configuration occurs when Y_i contains at least one non-crashed process, $X \subseteq Y_i$, $Y_i \cap X = \{\ell x\}$, and ℓx is the identity of the common representative of the non-crashed processes of X (or the identity of any of them if they all have crashed). In that configuration, any RESPONSE(id) message sent by any process $p_j \in Y_i$ carries an identity that belongs to L_i. It follows that $rec\_from_i \cap L_i \neq \emptyset$ (line 06), and so p_i does not issue *R_Broadcast* L_MOVE(L_i, Y_i) messages.

Proof of the upper wheel component. The proof is very similar to the proof of the lower wheel algorithm. Its structure is the same, and some of its parts are also the same. This is a direct consequence of the fact that both components are based on the same "wheel" principle. The proof considers an arbitrary run of the algorithm. As before, C denotes the set of processes that are correct in that run, and var_i^τ denotes the value of the local variable var_i of p_i at time τ.

```
Init: Y_i ← 𝒴[1]; L_i ← L_1^1
Task T3:
      (01) while true do
      (02)     Broadcast INQUIRY();
      (03)     wait until ( (∃j ∈ Y_i: a corresponding RESPONSE(id_j) is received from p_j)
      (04)                  ∨ φ-QUERY(Y_i) ); %  Y_i can dynamically change %
      (05)     let rec_from_i = {id_j received previously at line 03 };
      (06)     if (rec_from_i ≠ ∅) ∧ (rec_from_i ∩ L_i = ∅) then
      (07)               R_Broadcast L_MOVE(L_i, Y_i) end_if
      (08) end_do
Task T4: when L_MOVE(L_i, Y_i) is R_delivered: (L_i, Y_i) ← Next(L_i, Y_i)
Task T5: when INQUIRY() is received from p_j: send RESPONSE(repr_i) to p_j
Task T6: when trusted_i is read by the upper layer:
      (09) case  φ-QUERY(Y_i)  then return(min {j : j ∉ Y_i ∧ ¬φ-QUERY(Y_i ∪ {j})})
      (10)       ¬φ-QUERY(Y_i)  then return(L_i)
      (11) end_case
```

Figure 5: From $\diamond\phi^y + \diamond\mathcal{S}_x$ to Ω^z: upper wheel component (code for p_i)

LEMMA 4. *$\forall i \in C$, there are a pair (Λ_i, Υ_i) and a time τ_i such that $\forall \tau \geq \tau_i : (L_i^\tau, Y_i^\tau) = (\Lambda_i, \Upsilon_i)$.*

COROLLARY 2. *It exists a time after which no process sends* L_MOVE *messages.*

LEMMA 5. *$\forall i, j \in C : (\Lambda_i, \Upsilon_i) = (\Lambda_j, \Upsilon_j)$. (In the following, (Λ, Υ) denotes that pair.)*

THEOREM 3. *The sets $trusted_i$ implemented by the algorithm described in Figure 5 satisfy the property defining the class Ω^z.*

Proof Due to Lemma 5, there is a time after which all processes have permanently the same pair (Λ, Υ). We consider two cases:

- $\Upsilon \cap C = \emptyset$. In that case, due to the liveness property of the class $\diamond\phi^y$, there is a time after which any ϕ-QUERY(Υ) returns *true*. It follows then from line 09 that all the set $trusted_i$ are eventually equal and contain only the identity of a correct process (namely, the correct process with the lowest identity that does not belong to Υ).

- $\Upsilon \cap C \neq \emptyset$. In that case, any ϕ-QUERY(Υ) eventually always returns *false* (eventual safety property of the class $\diamond\phi^y$). It follows then from line 10 that all the sets $trusted_i$ are eventually permanently equal to Λ. As $|\Lambda| = z$, it remains to show that $\Lambda \cap C \neq \emptyset$.

  Let us assume for contradiction that $\Lambda \cap C = \emptyset$. Let p_i be a correct process. Due to properties ensured by the lower wheel (Theorem 2), there is a time after which any message RESPONSE($repr$) contains the identity of a correct process. From the assumption that Λ contains only faulty processes, it follows that there is a time τ_1 after which p_i cannot receive a RESPONSE message that carries the identity of a process belonging to Λ. Moreover, since set Υ contains at least one correct process, it follows from line 03-04 and the eventual safety property of the class $\diamond\phi^y$ that it exists a time τ_2 after which p_i always gets a RESPONSE($repr_j$) message from some process $p_j, j \in \Upsilon$ while waiting at lines 03-04. Finally, there is a time τ_i after which the predicate $(L_i, Y_i) = (\Lambda, \Upsilon)$ is permanently true (Lemma 4). Consequently, there is a time $\tau \geq \max(\tau_1, \tau_2, \tau_i)$ at which the predicate in the **if** statement of line 06 is not satisfied (i.e., at time τ, we have $rec\_from_i \neq \emptyset \wedge rec\_from_i \cap \Lambda = \emptyset$). It follows then that p_i broadcasts a L_MOVE(Λ, Υ) message. When p_i delivers such a message, it executes $(L_i, Y_i) \leftarrow$ Next(Λ, Υ). The fact that this occurs after the time τ_i contradicts Lemma 4.

$\square_{Theorem\ 3}$

A Particular Case. If $y = 0$, $\diamond\phi^y$ provides no information on failures. The upper wheel algorithm can be simplified by suppressing task $T6$ and line 04 of task $T3$. The value of $trusted_i$ is the current value of L_i.

5. LOWER BOUNDS AND REDUCIBILITY RESULTS

This section first states a lower bound related to the addition of failure detector classes (Fig. 2). It then proves the (ir)reducibility results stated in the grid of Figure 1.

5.1 A Lower bound when Adding $\diamond\mathcal{S}_x$ and $\diamond\phi^y$

This section shows that $(x + y + z > t + 1)$ is a lower bound when one wants to add failure detectors of the class $\diamond\mathcal{S}_x$ and $\diamond\phi^y$ to build a failure detector of the class Ω^z.

THEOREM 4. *Let us consider any system $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_x, \diamond\phi^y]$. $(\diamond\mathcal{S}_x + \diamond\phi^y \rightsquigarrow \Omega^z) \Leftrightarrow (x + y + z > t + 1)$.*

Proof [$\Leftarrow$ part] This part follows directly from the two wheels algorithm previously described in Sections 4.1 and 4.2.

[$\Rightarrow$ part] The proof of this part is by contradiction and considers the stronger system $\mathcal{AS}_{n,t}[\mathcal{S}_x, \phi^y]$. As $\mathcal{S}_x \subseteq \diamond\mathcal{S}_x$ and $\phi^y \subseteq \diamond\phi^y$, an impossibility result established in $\mathcal{AS}_{n,t}[\mathcal{S}_x, \phi^y]$ holds in $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_x, \diamond\phi^y]$.

Let us assume that there is an algorithm $\mathcal{T}$ that builds a failure detector of the class Ω^z in $\mathcal{AS}_{n,t}[\mathcal{S}_x, \phi^y]$ with $x + y + z \leq t + 1$. The contradiction is based on the following observations:

- Observation $O1$: Let f be the number of actual failures. When $f \leq t - y$, the only information that a

failure detector of the class ϕ^y can provide is the fact that the number of failures is $\leq t-y$.
Proof of O1. Consider a run where $f \leq t-y$. Let $E \subseteq \Pi$. Due to the triviality property of ϕ^y any ϕ-QUERY(E) returns *true* (resp., *false*) when $|E| \leq t-y$ (resp., $|E| > t$). As $f \leq t-y$ there is always a correct process in any set E such that $t-y < |E| \leq t$. It follows that, due the safety property of ϕ^y any ϕ-QUERY(E) returns *false* when $t-y < |E| \leq t$. Consequently the boolean value returned by any ϕ-QUERY(E) depends on the size of X, and does not depend on which processes define E. *End of the Proof of O1.*

- Observation *O2*: There is no algorithm that solves the k-set agreement problem in $\mathcal{AS}_{n,t}[\mathcal{S}_x]$ when $t \geq k+x-1$.
Proof of O2. This is a lower bound for solving the k-set agreement problem in $\mathcal{AS}_{n,t}[\mathcal{S}_x]$ established in [10]. *End of the Proof of O2.*

Let us now consider the transformation $\mathcal{T}$. In any run where $f \leq t-y$, it follows from *O1* that $\mathcal{T}$ can rely on ϕ^y only to know that the number of failures is $\leq t-y$. This implies that $\mathcal{T}$ can be used to build a failure detector of the class Ω^z in $\mathcal{AS}_{n,t-y}[\mathcal{S}_x]$. Moreover, it exists and algorithm $\mathcal{A}$ that solves the z-set agreement problem in $\mathcal{AS}_{n,t-y}[\Omega^z]$ (such an algorithm is presented in Section 3.1). Consequently, by combining transformation $\mathcal{T}$ and algorithm $\mathcal{A}$, one can solve the z-set agreement problem in $\mathcal{AS}_{n,t-y}[\mathcal{S}_x]$. Hence, it follows from *O2* that the constraint $t-y < z+x-1$ has to be satisfied, from which we obtain $x+y+z > t+1$: a contradiction. $\square_{Theorem\ 4}$

The following corollary is a consequence of Theorem 4.

COROLLARY 3. *The two wheels algorithm described in Figures 4 and 5 is optimal with respect to the possible values of x, y and z.*

As $\diamond\mathcal{S}_1$ (case $x=1$) and $\diamond\phi^0$ (case $y=0$) provide no information on failures, we directly obtain the following corollaries from the two wheel algorithm and Theorem 4.

COROLLARY 4. *It is possible to build a failure detector of the class Ω^z in $\mathcal{AS}_{n,t}[\phi^y]$ or $\mathcal{AS}_{n,t}[\diamond\phi^y]$ if and only if $y+z > t$.*

COROLLARY 5. *It is possible to build a failure detector of the class Ω^z in $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_x]$ if and only if $x+z > t+1$.*

5.2 Relations between $\mathcal{S}_x/\diamond\mathcal{S}_x$ and $\phi^y/\diamond\phi^y$

THEOREM 5. *Let $1 \leq x \leq t+1$ and $1 \leq y \leq t$. It is not possible to build a failure of the classes ϕ^y ,$\diamond\phi^y$ neither in $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_x]$ nor in $\mathcal{AS}_{n,t}[\mathcal{S}_x]$.*

Proof The proof considers the "stronger" system $\mathcal{AS}_{n,t}[\mathcal{S}_x]$. the proof remains valid for a system $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_x]$, since $\mathcal{S}_x \subseteq \diamond\mathcal{S}_x$. Similarly, as $\phi^y \subseteq \diamond\phi^y$ the proof considers only the "weaker" class $\diamond\phi^y$ in the following. The proof is by contradiction. Let us assume that there is a failure detector $\mathcal{F}$ of the class $\mathcal{S}_x$ and an algorithm $\mathcal{A}$ that transforms $\mathcal{F}$ into a failure detector of the class $\diamond\phi^y$. We exhibit a run R in which the eventual safety property of the class $\diamond\phi^y$ is not satisfied.

Let $E \subseteq \Pi$, $|E| = t-y+1$ and $E \cap C \neq \emptyset$. Let p_c be a correct process that does not belong to set E. Moreover, p_c is never suspected by $\mathcal{F}$ in run R. Let τ_0 be the time at which any ϕ-QUERY(E) invoked after time τ_0 returns the value *false*. Such a time exists due to the correctness of algorithm $\mathcal{A}$ and the eventual safety property of the class $\diamond\phi^y$. We consider the following runs $R1$ and $R1'$:

- Runs $R1$ and R are indistinguishable by all processes until time τ_0. A time τ_0+1, all processes that belong to E crash. Let $\tau_1 > \tau_0$ be a time at which a process $p_i \in \Pi - E$ invokes ϕ-QUERY(E) and obtains the value *true*. Such a time must exist due to liveness property of the class $\diamond\phi^y$.

- Runs $R1'$ and R are indistinguishable by all processes until time τ_0. In the run $R1'$, all the processes in E are correct, but all the messages they send between times τ_0+1 and τ_1 are delayed until time τ_1+1.

Moreover, both runs are such that the outputs of the failure detector $\mathcal{F}$, at each process, are exactly the same between the times 0 and τ_1. (Let us notice that whatever the output of $\mathcal{F}$ in $R1$, the output of $\mathcal{F}$ can be exactly the same in $R1'$ without violating the properties of the class $\mathcal{S}_x$. As p_c is correct in $R1$ and $R1'$ and never suspected in $R1$ and $R1'$, limited scope perpetual accuracy is insured. Since strong completeness is an eventual property, it is always satisfied in any finite prefix of any execution.) Clearly, up to time τ_1, the processes that belong to $\Pi - E$ cannot distinguish the run $R1$ from the run $R1'$. It follows that, in the run $R1'$, an invocation of ϕ-QUERY(E) by p_i at time $\tau_1 > \tau_0$ returns the value *true*. But in run $R1'$, a ϕ-QUERY(E) issued after time τ_0 must return the value *false*: a contradiction. $\square_{Theorem\ 5}$

THEOREM 6. *Let $0 \leq y < t$ and $1 < x \leq t+1$. It is not possible to build a failure of the class $\mathcal{S}_x$ or $\diamond\mathcal{S}_x$ neither in $\mathcal{AS}_{n,t}[\diamond\phi^y]$ nor in $\mathcal{AS}_{n,t}[\phi^y]$.*

Proof Let us first notice that we need to prove only the impossibility to build a failure detector of the class $\diamond\mathcal{S}_x$ in $\mathcal{AS}_{n,t}[\phi^y]$. The proof is by contradiction and uses the following observations.

Observation *O1*: Let f be the number of actual failures. When $f \leq t-y$, the only information that a failure detector of the class ϕ^y can provide is the fact that the number of failures is $\leq t-y$. (This observation has already been stated and proved in Theorem 4.)

Observation *O2*: There are algorithms that solve the k-set agreement problem in $\mathcal{AS}_{n,t}[\mathcal{S}_x]$. All these algorithms require $t \leq k+x-2$. (Examples of such algorithms can be found in [10, 18]. The lower bound on t is established in [10].)

Observation *O3*: The k-set agreement problem can be solved in $\mathcal{AS}_{n,t-y}[\emptyset]$ if and only if $k > t$. (The proof of this observation constitutes an important result of fault-tolerant distributed computing. It can be found in [1, 11, 22].)

Let us suppose that there is an algorithm $\mathcal{A}$ that builds a failure detector of the class $\diamond\mathcal{S}_x$ from a failure detector of the class ϕ^y. In any run where $f \leq t-y$, it follows from *O1* that $\mathcal{A}$ can rely on ϕ^y only to know that the number of failures is $\leq t-y$. Consequently, $\mathcal{A}$ can build a failure detector of the class $\diamond\mathcal{S}_x$ in a system $\mathcal{AS}_{n,t-y}[\emptyset]$. This means that one can use $\mathcal{A}$ to solve the the $(t-y)-x+2$-set agreement problem using any algorithm listed in observation *O2* in a system

$\mathcal{AS}_{n,t-y}[\emptyset]$. We then conclude from $O3$ that $(t-y)-x+2 > t-y$, i.e., $x \leq 1$, a contradiction with the assumption $1 < x \leq n$ [1]. $\square_{Theorem\ 6}$

5.3 From Ω^z to $\phi^y/\diamond\phi^y$ or $\mathcal{S}_x/\diamond\mathcal{S}_x$

It has been shown (Corollaries 5 and 4) that it is possible to build a failure detector of the class Ω^z from any failure detector of the classes $\mathcal{S}_x/\diamond\mathcal{S}_x$ (resp., $\phi^y/\diamond\phi^y$) if and only if $x+z > t+1$ (resp., $y+z > t$). This section shows that it is not possible to build a failure detector of the classes $\mathcal{S}_x/\diamond\mathcal{S}_x$ (resp., $\phi^y/\diamond\phi^y$) from any failure detector of the class Ω^z. The proofs of these impossibilities derived from Theorem 6 and 5.

THEOREM 7. *Let $1 \leq y \leq t$ and $1 \leq z \leq t+1$. It is impossible to build a failure detector of a class $\phi^y/\diamond\phi^y$ in $\mathcal{AS}_{n,t}[\Omega^z]$.*

Proof The proof is by contradiction. Let us assume that there is an algorithm $\mathcal{A}$ that builds a failure detector of a class $\diamond\phi^y$, $1 \leq y \leq t$, from any failure detector of a class Ω^z, $1 \leq z \leq t+1$. Due to Corollary 5, it is possible to build a failure detector of a class Ω^z in $\mathcal{AS}_{n,t}[\diamond\mathcal{S}_x]$ when $x+z > t+1$, i.e., when $z > t-x+1$. Combining this construction with the algorithm $\mathcal{A}$ we obtain an algorithm $\mathcal{B}$ that builds a failure detector of the class ϕ^y, $1 \leq y \leq t$ from a failure detector of the class $\diamond\mathcal{S}_x$ when $t+1-z < x \leq t+1$ and $1 \leq z \leq t+1$. But such an algorithm $\mathcal{B}$ contradicts Theorem 5 that states that there is no such algorithm when $1 \leq x \leq t+1$ and $1 \leq y \leq t$. $\square_{Theorem\ 7}$

THEOREM 8. *Let $1 < x, z \leq t$. It is impossible to build a failure detector of the class $\mathcal{S}_x/\diamond\mathcal{S}_x$ in $\mathcal{AS}_{n,t}[\Omega^z]$.*

Proof The proof is similar to the proof of Theorem 7. It is left to the reader. $\square_{Theorem\ 8}$

5.4 Optimality in the Grid

It follows from all the previous theorems and lemmas that, when we consider all the failure detector classes depicted in Figure 1, Ω^k is the weakest class that allows solving the k-set agreement problem. This constitutes a first step towards the characterization of the weakest failure detector class for solving that problem.

6. REFERENCES

[1] Borowsky E. and Gafni E., Generalized FLP Impossibility Results for t-Resilient Asynchronous Computations. *Proc. 25th ACM Symp. on the Theory of Computing (STOC'93)*, ACM Press, pp. 91-100, 1993.

[2] Chandra T., Hadzilacos V. and Toueg S., The Weakest Failure Detector for Solving Consensus. *Journal of the ACM*, 43(4):685–722, 1996.

[3] Chandra T.D. and Toueg S., Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2):225-267, 1996.

[4] Chaudhuri S., More *Choices* Allow More *Faults:* Set Consensus Problems in Totally Asynchronous Systems. *Information and Computation,* 105:132-158, 1993.

[5] Chu F., Reducing Ω to $\diamond\mathcal{W}$. *Information Processing Letters*, 76(6):293-298, 1998.

[6] Delporte-Gallet C., Fauconnier H. and Guerraoui R., (Almost) All Objects are Universal in Message Passing Systems. *Proc. 19th Symp. on Distributed Computing (DISC'05)*, Springer Verlag LNCS #3724, pp. 184-198, 2005.

[7] Fischer M.J., Lynch N. and Paterson M.S., Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374-382, 1985.

[8] Guerraoui R. and Raynal M., The Information Structure of Indulgent Consensus. *IEEE Transactions on Computers.* 53(4), 53(4):453-466, 2004.

[9] Hadzilacos V. and Toueg S., Reliable Broadcast and Related Problems. In *Distributed Systems*, ACM Press, New-York, pp. 97-145, 1993.

[10] Herlihy M.P. and Penso L. D., Tight Bounds for k-Set Agreement with Limited Scope Accuracy Failure Detectors. *Distributed Computing*, 18(2): 157-166, 2005.

[11] Herlihy M.P. and Shavit N., The Topological Structure of Asynchronous Computability. *Journal of the ACM*, 46(6):858-923, 1999.

[12] Lamport L., The Part-Time Parliament. *ACM Transactions On Computer Systems*, 16(2):133-169, 1998.

[13] Mostefaoui A., Rajsbaum S. and Raynal M., Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems. *Journal of the ACM,* 50(6):922-954, 2003.

[14] Mostefaoui A., Rajsbaum S. and Raynal M., The Combined Power of Conditions and Failure Detectors to Solve Asynchronous Set Agreement. *Proc. 24th ACM Symp. on Principles of Distributed Computing (PODC'05)*, ACM Press, pp. 179-188, 2005.

[15] Mostefaoui A., Rajsbaum S., Raynal M. and Travers C., From $\diamond\mathcal{W}$ to Ω: a Simple Bounded Quiescent Reliable broadcast-based Transformation. *Tech Report 1759*, IRISA, University of Rennes 1 (France), 2005.

[16] Mostefaoui A., Rajsbaum S., Raynal M. and Travers C., Irreducibility and Additivity of Set Agreement-oriented Failure Detector Classes. *Tech Report 1758*, IRISA, University of Rennes 1 (France), 2005. ftp://ftp.irisa.fr/techreports/2005/PI-1758.ps.gz.

[17] Mostefaoui A. and Raynal M., Solving Consensus Using Chandra Toueg's Unreliable Failure Detectors: a General Quorum Based Approach. *Proc. 13th Symp. on Distributed Computing (DISC'99)*, Springer Verlage LNCS #1693, pp. 49-63, 1999.

[18] Mostefaoui A. and Raynal M., k-Set Agreement with Limited Accuracy Failure Detectors. *Proc. 19th ACM Symp. on Principles of Distributed Computing (PODC'00)*, ACM Press, pp. 143-152, 2000.

[19] Mostefaoui A. and Raynal M., Leader-Based Consensus. *Parallel Processing Letters*, 11(1):95-107, 2001.

[20] Neiger G., Failure Detectors and the Wait-free Hierarchy. *Proc. 14th ACM Symp. on Principles of Distributed Computing (PODC'95)*, ACM Press, pp. 100-109, 1995.

[21] Raynal M., A Short Introduction to Failure Detectors for Asynchronous Distributed Systems. *ACM SIGACT News, Distributed Computing Column*, 36(1):53-70, 2005.

[22] Saks M. and Zaharoglou F., Wait-Free k-Set Agreement is Impossible: The Topology of Public Knowledge. *SIAM Journal on Computing*, 29(5):1449-1483, 2000.

[23] Schiper A., Early Consensus in an Asynchronous System with a Weak Failure Detector. *Distributed Computing*, 10:149-157, 1997.

[24] Yang J., Neiger G. and Gafni E., Structured Derivations of Consensus Algorithms for Failure Detectors. *Proc. 17th ACM Symp. on Principles of Distributed Computing (PODC'98)*, pp.297-308, 1998.

[1] Let us remind that the failure detectors of the classes $\mathcal{S}_1$ and $\diamond\mathcal{S}_1$ provide no information on failures.