International Journal of General Systems

Towards a categorical model for resilience

Submission ID	248868502
Article Type	Research Article
Keywords	Category theory, resilience, multiplicity principle , drones
Authors	Erwan Beurier, Dominique Pastor, Nora Boulahia-Cuppens, Frédéric Cuppens

Towards a categorical model of resilience

Erwan Beurier [†,a,b] [*], Dominique Pastor[c], Nora Boulahia-Cuppens[a] and Frédéric Cuppens[a]

[a] *Département de Génie Informatique et Génie Logiciel, Polytechnique Montréal, Montréal, Québec, Canada;*
[b] *IRT SystemX, 2 boulevard Thomas Gobert, 91120 Palaiseau, France;*
[c] *Mathematical and Electrical Engineering Department, IMT Atlantique, Lab-STICC, Université de Bretagne Loire, Technopôle Brest-Iroise, CS 83818 - 29238 Brest cedex 03, France*

July 3, 2024

Abstract

We link resilience of systems to the biological property of degeneracy. Degeneracy is the property of having backup subsystems in case of an incident, and has been studied by biologists and mathematicians. A system can be represented in category theory, in which degeneracy becomes the Multiplicity Principle.

However, in a category, any two consecutive arrows need to compose. This leads to the existence of arrows that do not interpret semantically, which may "hide" the MP under a flood of arrows.

In this paper, we define an extension to Ehresmann and Vanbremeersch's categorical model to solve this issue and thus, better describe systems and their properties, resilience being the main focus. This extension consists of a decomposition of each category into subcategories, that we call layers, each with its own semantics.

We then use this modelling to guide us towards endowing a simulated swarm of drones with resilience.

Keywords: Category theory, resilience, Multiplicity Principle, drones.

Contents

[*] † Corresponding author. Email: erwan.beurier@polymtl.ca

1 Introduction

A complex system displays, by definition, a high number of interacting components. Most of the time, it also shows many functionalities, or its main functionality can be divided in several smaller ones. Perhaps a specific system functionality is of interest because it confers a particular property. Therefore, it is necessary to study all this data simultaneously, thence the need to extract and isolate this functionality to gain a clear understanding of its purpose and implementation.A decomposition would thus be useful. One such property could be resilience.

Resilience is a widely-spread concept, used in many areas of science: ecology [1], psychology [2], sociology [3], cybersecurity [4, 5], among others. The main idea remains the same: a system is called resilient when it keeps functioning in an acceptable way ("acceptable" being context-dependent) even if said system is damaged or under attack, and is capable of a rebound to bring it back to its original state. It is a reduced functionality mode inherent to any living being and even broader systems, like social systems or multi-agent systems.

Resilience has become a property of interest in engineering, particularly for critical systems and infrastructures - those systems that cannot stop without endangering lives, or at least the comfort, of one or more people while they are in use. Think of a computer, a submarine, a boat, or a plane in operation.

Degeneracy is a concept that originates from biology [6]. It is defined as the ability of a biological system to present two (or more) structurally different subsystems that perform the same function. More precisely, the two subsystems should perform different functions, and share at least one. The main studied consequence of degeneracy is the emerging complexity of the system [6].

As resilience seems to happen in a system only when critical functionalities are performed by backup subsystems, it is natural to link degeneracy and resilience. In this article, we argue that degeneracy must be coupled with two other functionalities in order to achieve resilience.

Our main tool to carry this study, is category theory. Category theory is a subfield of abstract algebra that already proved useful in biology [7, 8]. [9] even proposed a complete representation of complexity and emergence in terms of category theory based on a biological background. The categorical translation of degeneracy is the Multiplicity Principle (MP), and it is the case in point for the necessity of having a decomposition. The MP has several categorical implications, in particular in relation to the emergence of colimits in a sequence of categories [9, Theorem 3, Chapter 4, Section 6.1, pages 133-134]. Hence the description of complexity in terms of categories.

In this paper, the goal is to present an extension of the Evolutive Systems introduced by [10]. However, most of the technical details, mathematical justifications will be put in appendix in order to make the paper lighter.

Section 2 is a preamble. We start with the basics of category theory for biology and the representation of systems. We also quickly tackle some foundational issues for the mathematician reader. We conclude this section with the definition of resilience that we will use throughout this article. If the reader is not familiar with category theory, an introductory section may be found in Appendix A.

In Section 3, we describe a new modelling of systems using category theory, based on this vision of a superposition of thinner categories. A system is described as a category, and this category can decompose into smaller categories whose superposition gives the base category. The main tool is the colimit of categories, which is seen as a superposition under certain circumstances.

We need this decomposition because the MP, as originally described, suggests a form of redundancy that implies the existence of two structurally different subsystems performing exactly the same functions. This is not the intended meaning of degeneracy. Degeneracy implies the existence of two organs, that have their own functions, but share one. If one layer represents a point of view of the system restricted to a certain functionality, we allow for the MP to happen in the layers, rather than in the colimit category, which gives a better translation of degeneracy in category theory.

Using the definition of "resilience" given in Section 2.2, we extend the above-mentioned modelling to set up for a more precise study of resilience. A system will be divided into several parts, each contributing to a certain function or to the different capacities of resilience.

Finally, in Section 4, we show how our framework lays the foundations for the implementation of resilience. We use this modelling to describe and study the properties of resilience of an example system consisting of a swarm of drones. We also report a simulation to validate this theory.

2 Categories, and resilience

As mentioned in the introduction, our work uses category theory as a framework for describing systems. In case the reader is not familiar with category theory, a brief introductory crash course can be found in Appendix A. Further references in the topic include [11], [12], [13] for a scientific audience or [14] for mathematicians.

In this section, we introduce the modelling of biological systems by category theory (Section 2.1). We end this section with a brief discussion on the definition of resilience that we will use in this paper (Section 2.2).

Notation. Categories will be denoted by cursive letters $\mathscr{C}, \mathscr{D}, \mathscr{E}, \ldots, \mathscr{P}$. Functors are written with the letters F, F' and diagrams are written with the letters $D : \mathscr{D} \to \mathscr{C}$, $E : \mathscr{E} \to \mathscr{C}$. Diagrams to $\boldsymbol{Cat}$ will be noted as $P : \mathscr{P} \to \boldsymbol{Cat}$. Comma-categories will be denoted $(D \downarrow E)$. The diagonal functor will be denoted $\Delta^{\mathscr{C}}_{\mathscr{X}} : \mathscr{X} \to \boldsymbol{Func}(\mathscr{C}, \mathscr{X})$.

We denote by $\boldsymbol{Cat}$ (and $\boldsymbol{PartCat}$) the categories of small categories and functors (partial functors, respectively). Note that $\boldsymbol{Cat}$ is not a small category itself, and for our purposes, we need a small category of categories containing finitely-generated categories (further explanation can be found in Appendix A.3).

Hyperstructures and categories. Other mathematical tools exist to carry out this study, most notably hyperstructures [15, 16, 17, 18, 19]. They are used to describe the multi-level complexity of systems in terms of nested sets, of sets, of sets... with links and relations. [20, Section 4, pages 564-566] compares both tools. Hyperstructures are a more general framework than category theory, in the sense that everything written in categories can be translated to hyperstructures. Hyperstructures are however not part of a more general theory, say "hyperstructure theory". Consequently, fewer tools, theorems and references are available, making their use more difficult.

2.1 Categories and biology

Evolutive systems. [9] used categories to model biological, cognitive, and social systems. A system is seen as the category generated by its graph of interactions between components. The category is then quotiented by a handmade equivalence relation that translates the equivalence between different sequences (compositions) of interactions.

For clarity, a system is modelled by a category whose objects are the components of the systems, and whose arrows are the interactions, functionalities of a component onto another one.

Each component can be decomposed into the system of its own components, e.g. a CPU can be decomposed into the system of the different transistors, electronic components, connections, cache memory, etc. A system can be decomposed even further depending on the goal of the modeller.

Each component interacts with other components, whether by receiving input, sending output, or more abstract interactions such as decomposition of a component into smaller parts. These interactions are represented as arrows in the category representing the system.

A system evolves through time, so we cannot model a system with only one category. [9] proposed the notion of Evolutive System:

Definition 2.1 (Evolutive System)**.** An *evolutive system* is a functor $\mathbb{E} : (T, \leqslant) \to \boldsymbol{PartCat}$ such that T is a subset of the real numbers $T \subset \mathbb{R}$ equipped with the usual comparison of numbers.

In this definition, T represents a timescale, the life span of the system. Note that it doesn't need to be a connected subset of $\mathbb{R}$, even if it would not have much physical sense otherwise. For all $t \in T$, $\mathbb{E}(t)$ is a category representing the configuration of the system at instant t, and for $t \leqslant t'$, $\mathbb{E}(t, t')$ is a partial functor describing the evolution of the system between t and t'.

A partial natural transformation $\mathbb{E} \to \mathbb{E}'$ between two evolutive systems, is called an evolutive functor in [9].

Definition 2.2 (Category of evolutive systems)**.** Evolutive systems and partial natural transformations (Definition A.10) define a category $\boldsymbol{ES}$.

Clusters. Considering the configuration $\mathbb{E}(t)$ at time t of an evolutive system $\mathbb{E} : (T, \leqslant) \to \boldsymbol{PartCat}$, a diagram $D : \mathscr{D} \to \mathbb{E}(t)$ gathers a set of components and some interactions. If such diagram has a colimit cD in $\mathbb{E}(t)$, then we interpret this as follows. The components in $\operatorname{Im}(D)$ (the image of D) "assemble" into the component cD. Or, if we zoom in on the component cD, we see its internal structure as the image of D. We say that such a diagram D *decomposes* the corresponding component cD.

Given two diagrams with colimits in the category, we can generally observe sets of arrows from the first diagram to the second diagram. Some of those sets may have several properties, we call them clusters, and they play a role in the study of degeneracy.

We introduce the following notation. We consider two diagrams $D : \mathscr{D} \to \mathscr{C}$ and $E : \mathscr{E} \to \mathscr{C}$. If G is a subset of the objects of the comma-category $(D \downarrow E)$ (i.e. G is a set of triples $(d, g : D(d) \to E(e), e)$; Definition A.3), we define $G(d)$ as:

$$G(d) = \{(d,g,e) \in G\}$$

In other words, $G(d)$ is the subset of G whose first element is the fixed d.

For the sake of simplicity, we write $g : D(d) \to E(e) \in (D \downarrow E)$ instead of the whole triple $(d, g : D(d) \to E(e), e)$.

Using this notation, we define a cluster.

Definition 2.3 (Cluster). Let $D : \mathscr{D} \to \mathscr{C}$ and $E : \mathscr{E} \to \mathscr{C}$ be two diagrams in a category $\mathscr{C}$. A *cluster* is a subset of the objects of the comma-category $(D \downarrow E)$ that verifies:

(CLU-1) for all $d \in \mathscr{D}$, there exists $g : d \rightharpoonup e \in G$ for some $e \in \mathscr{E}$ i.e. $G(d) \neq \varnothing$.

(CLU-2) for all $d \in \mathscr{D}$, if $g : d \rightharpoonup e \in G$ and $g' : d \rightharpoonup e' \in G$, then they are connected in the comma category $(D(d) \downarrow E)$

(CLU-3) if $a : d' \to d \in \mathscr{D}$ and $g : d \rightharpoonup e \in G(d)$, then $gD(a) : d' \rightharpoonup e \in G(d')$, i.e. $\{gD(a) \mid g \in G, a : d' \to d\} \subset G(d')$

(CLU-4) G is maximal for the inclusion $\subseteq$ among the subsets $(D \downarrow E)$ that verify (CLU-1), (CLU-2) and (CLU-3)

Equivalently [21, Proposition 3.11, Section 3, page 502], a cluster $G : D \to E$ is an element of the set:

$$\underset{d \in \mathscr{D}}{\mathrm{Lim}}\, \underset{e \in \mathscr{E}}{\mathrm{Colim}}\, \mathrm{Hom}_{\mathscr{C}}(D(d), E(e))$$

If $G : D \to D'$ and $G' : D' \to D''$ are two consecutive clusters, then they can compose according to the following law: the composition $G' \circ G$ is the cluster generated by the set $\{g' \circ g \mid g \in G, g' \in G'\}$. Such a set is generally not a cluster, however it generates a unique cluster [21, Lemma 3.6, Section 3, page 499]. We thus have:

Definition 2.4 (Category of clusters). Let $\mathscr{C}$ be a (small) category. The *category of clusters of* $\mathscr{C}$, denoted $\mathrm{Clu}(\mathscr{C})$, is the category consisting of:

Objects: Objects are small diagrams $D : \mathscr{D} \to \mathscr{C}$

Morphisms: Arrows are clusters $D \to E$

Identities: Identities $D \to D$ are the clusters generated by the set of identities:

$$\left\{\mathrm{id}_{D(d)} : D(d) \to D(d) \mid d \in \mathscr{D}\right\}$$

Composition: The composition of $G : D \to D'$ with $G' : D' \to D''$ is the cluster generated by the set $\{g' \circ g \mid g \in G, g' \in G'\}$

Note that cocones are special cases of clusters. In fact, a cocone $D \to A$ is exactly a cluster of the form $D \to \Delta(A)$. Thus, it is natural to compose clusters and cocones:

Definition 2.5 (Composition of a cluster with a cocone). Let $G : D \to E$ be a cluster and let $\alpha : E \to \Delta(A)$ be a cocone from E to A.

The *composition of* α *with* G, denoted by $\alpha \circ G$, is the cocone $\alpha \circ G : D \to \Delta(A) = \{\alpha_e \circ g \mid g : D(d) \to E(e) \in G, \alpha_e : E(e) \to A\}$.

Note that the result of the composition of a cluster with a cocone is also a cocone.

Multiplicity Principle. Consider now two diagrams D and E with colimits $\mathrm{Colim}\,(D) : D \to \Delta\,(cD)$ and $\mathrm{Colim}\,(E) : E \to \Delta\,(cE)$. Also assume that there is a cluster $D \to E$. The composite $\mathrm{Colim}\,(E) \circ G$ is a cluster $D \to \Delta\,(cE)$; equivalently, it is a cocone from D to cE. Thus, there is a unique arrow $g : cD \to cE$ due to the UMP of cD. Hence the following notion:

Definition 2.6 (Binding of a cluster)**.** Let $G : D \to E$ be a cluster between two diagrams D and E that both have a colimit (hereafter denoted by $\mathrm{Colim}\,(D) : D \to \Delta\,(cD)$ and $\mathrm{Colim}\,(E) : E \to \Delta\,(cE)$).

The *binding of* G is the unique arrow $g : cD \to cE \in \mathscr{C}$ such that $\Delta\,(g) \circ \mathrm{Colim}\,(D) = \mathrm{Colim}\,(E) \circ G$; that is, the following square commutes (in $\mathrm{Clu}\,(\mathscr{C})$):

$$\begin{tikzcd}
\Delta\,(cD) \arrow[r, "\Delta(g)"] & \Delta\,(cE) \\
D \arrow[u, "\mathrm{Colim}(D)"] \arrow[r, "G"'] \arrow[ur, phantom, "\checkmark"] & E \arrow[u, "\mathrm{Colim}(E)"']
\end{tikzcd}$$

We equivalently say that G *binds to* g. We say that G *binds to an isomorphism* when g is an isomorphism.

With the terminology introduced just above, whenever two diagrams have a colimit, if there is a cluster between those two diagrams, then the binding of the cluster always exists.

Remark 2.7. Note that $\Delta\,(g)$ is a cluster $\Delta\,(cD) \to \Delta\,(cE)$, so the previous diagram really is a diagram in $\mathrm{Clu}\,(\mathscr{C})$.

[9, discussion below definition, Part A, Chapter 3, Section 1.4, page 79] introduced the following functor, but it is left unnamed.

Definition 2.8 (Cluster-composition functor)**.** Let $G : D \to E$ be a cluster. We define the *cluster-composition functor* ΩG as:

$$\Omega G : \begin{cases} \textbf{\textit{Cocones}}\,(E) & \longrightarrow & \textbf{\textit{Cocones}}\,(D) \\ \alpha & \longmapsto & \alpha \circ G \\ u & \longmapsto & u \end{cases}$$

The cluster-composition functor is a specific functor that preserves the "peaks" of cocones. However, not all functors that preserve peaks are a cluster-composition functor.

Definition 2.9 (Multiplicity Principle)**.** Let $D : \mathscr{D} \to \mathscr{C}$ and $E : \mathscr{E} \to \mathscr{C}$ be two diagrams to $\mathscr{C}$. We say that D *and* E *verify the Multiplicity Principle* (or MP for short) when:

(MP-1) There exists an isomorphic functor $I : \textbf{\textit{Cocones}}\,(E) \to \textbf{\textit{Cocones}}\,(D)$ that preserves peaks;
(MP-2) For all $G : D \to E$ or $G : E \to D$, ΩG is not an isomorphism.

In other words: $\mathscr{C}$ verifies the Multiplicity Principle when there are two diagrams D and E whose cocone categories are isomorphic, but such that for all cluster $G : D \to E$, the composition of cocones by G does not define an isomorphism (there may also be no cluster at all). One may also say: the cocone categories are isomorphic, but ΩG is not the witnessing isomorphism.

The Multiplicity Principle was first introduced in [22], and only for diagrams with the same colimit. The cluster-composition functor was first introduced in [9, Part A, Chapter

3, Section 4.3, page 91] in order to generalise the original MP to diagrams without colimits. Also note that [23, Part III, Chapter 6, Definition 6.2.9, page 178] introduces a lax and strict form of the MP. The strict form requires that the peaks of the cocones must be preserved by the isomorphism, which the lax form does not require. Actually, in the context of the representation of systems, only the strict MP makes sense. Although the MP introduced in [24] was meant to be lax, it resulted strict in the preorder of tests. In the present work, we focus on the strict MP, so by default, we will drop the "strict" mention.

Proposition 2.10. *If E' is a subdiagram of E (meaning that* $\mathrm{dom}(E') \subset \mathrm{dom}(E)$*), such that there is an isomorphism* $\boldsymbol{Cocones}(E) \to \boldsymbol{Cocones}(E')$ *that preserves peaks, then the MP does not hold between E' and E.*

Proof. If E' is a subdiagram of E, then we at least have the cluster generated by all identities of the objects in $\mathrm{dom}(E')$:

$$G = \left\{ \mathrm{id}_{e'} : E(e') \to E(e') \mid e' \in \mathrm{dom}(E') \right\}$$

The corresponding cluster-composition functor defines an isomorphism $\boldsymbol{Cocones}(E) \to \boldsymbol{Cocones}(E')$ that preserves peaks. □

In the context of the representation of systems, this proposition can be interpreted as follows: even if E' is a subsystem of E with essentially the same functionality, they are not independent enough. As we will see later, the MP requires some independence between E' and E.

2.2 What is resilience, really?

Resilience is generally defined as the ability of a system to keep working somewhat properly even when damaged (to a certain point). [25, Table 1, pages 93-94] reviews several definitions of resilience from several fields and writes a general definition that encompasses all of them. For [25], a general and reliable definition of resilience boils down to these three capacities:

1. Absorptive capacity; this is the ability to reduce the impact of perturbations with little effort.
2. Adaptive capacity; once the perturbation happened, this is the ability to internally modify the system and/or its parameters to ensure that the main functions are still performed.
3. Recoverability; once the perturbation is over, this is the ability to get the system back to an acceptable (or even better) performance.

While a significant number of authors focus their study of resilience on the second capacity (that of adaptivity), [25] offer a generalist definition of resilience that takes every aspect into account.

Note that there seems to be a chronology with these three capacities, which would turn them into steps: first the system tries to absorb the perturbation (absorption step), then it adapts to the damage done by said perturbation (adaptation step), and finally it recovers from it (recovery step).

Besides, even if we mention "resilience" without much more details, we always refer to the resilience of a certain function of the system under study. By no means do we consider a system that is perfect and indestructible in every aspect. We insist that resilience is also a limited property. A system cannot undergo infinite perturbations. A system can only be resilient for a certain portion of time, after which any further damage may be fatal.

From now on, we consider this three-step (or three-capacity) definition of resilience. In particular, in Section 3.3, we translate each step into categorical structures. This will be illustrated by an example in Section 4.

3 Using superpositions to model systems

The Multiplicity Principle (MP) was introduced in [26] as a categorical translation of degeneracy as described in [6]. However, there is a loss of information in the process. Degeneracy implies that the two subsystems have several functions, one of which is shared by both. In the current state of the MP, the two subsystems have *exactly* the same functions.

In this paper, we consider the following solution (see Definition 3.1 for a more formal presentation). A system is seen as a superposition of layers. Each layer describes the architecture of the system, i.e. all components and interactions, that contribute to a given function. In a sense, a system is a superposition of all its functions. The decomposition of a system into its functions depends on the choice of the engineer using the model.

The main idea is that, if two subsystems have several functions, and one is shared by both, then each subsystem should appear in several layers, both should appear in one specific layer (that of the shared function), and this layer should verify the MP. However, once these layers are superposed, they translate the idea of two subsystems that have several functions and share one, which is the idea behind degeneracy.

It is now time to introduce how to model a system with categories, and in particular, resilience, using the tools we developed in the previous section.

3.1 Layered system

Before including the basis for resilience, we consider systems independently on whether they have resilience or not. This definition is to be seen as an extension of the Evolutive Systems from [9]. For the purpose of this article, we do not consider the hierarchical part of ES in order to keep things simple. However, with some meticulousness, we can add hierarchy to the components of the systems.

For the sake of simplicity, we use the category of categories $\boldsymbol{Cat}$. For foundational issues that may arise, the reader may refer to Appendix A.3.

Definition 3.1 (Layered system)**.** A *layered system*, or simply *system*, is a tuple $\mathscr{S} = (\mathscr{C}, P : \mathscr{P} \to \boldsymbol{Cat})$ where:

1. $\mathscr{C} \in \boldsymbol{Cat}$ is a category
2. $P : \mathscr{P} \to \boldsymbol{Cat}$ is a diagram
3. $\mathscr{P}$ has an initial element denoted by p_0
4. $P(p_0) = |\mathscr{C}|$, that is, $P(p_0)$ is the subcategory of the objects of $\mathscr{C}$
5. for all $p \in \mathscr{P}$, $\mathrm{Ob}(P(p)) = \mathrm{Ob}(\mathscr{C})$
6. for all $a : p \to p' \in \mathscr{P}$, $\mathrm{Ob}(P(a)) : \mathrm{Ob}(P(p)) \to \mathrm{Ob}(P(p'))$ is the identity and $\mathrm{Mor}(P(a)) : \mathrm{Mor}(P(p)) \hookrightarrow \mathrm{Mor}(P(p'))$ is an injection
7. $\mathrm{Colim}(P) \cong \mathscr{C}$

The idea is as follows. The category $\mathscr{C}$ represents the whole system at a given instant t. The objects of $\mathscr{C}$ are the components of the system, and its arrows are all the interactions worth studying. This category is basically the same as the *configuration* described in [9, Part A, Chapter 1, Sections 3-4, pages 32-42], and a collection of such categories makes an Evolutive System (Definition 2.1).

However, the configuration alone felt too confusing for a practical study such as the examples we see in Section 4. As any two consecutive arrows need to compose in a category, if those two arrows have two incompatible practical interpretations, the interpretation of the composite is not clear. Thus the necessity for splitting the category in several layers, each layer allowing only the composition between two arrows that have a compatible interpretation. This decomposition finds its mathematical justification in the fact that the colimit of diagrams $D : \mathscr{D} \to \boldsymbol{Cat}$ can be seen as a superposition of the categories $D(d)$'s if the arrows $D(a) : D(d) \to D(d')$ preserve the objects. In particular, every category can be seen as a superposition of preorders.

The diagram $P : \mathscr{P} \to \boldsymbol{Cat}$ is thus a decomposition of the system into simpler parts. The use of this diagram will be made clearer in the example in Section 4.

Definition 3.2 (Morphism of systems)**.** Consider two systems $(\mathscr{C}, P : \mathscr{P} \to \boldsymbol{Cat})$ and $(\mathscr{C}', P' : \mathscr{P} \to \boldsymbol{Cat})$.

A *morphism of systems* $(\mathscr{C}, P) \to (\mathscr{C}', P')$ is a pair (S, σ) such that:

1. $S : \mathscr{C} \to \mathscr{C}'$ is a partial functor
2. $\sigma : P \to P'$ is a partial natural transformation (see Definition A.10)
3. For all $a : p \to p' \in \mathscr{P}$, the following diagram commutes:

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "S"] & \mathscr{C}' \\
P(p) \arrow[u, "\mathrm{Colim}(P)_p"] \arrow[r, "\sigma_p"'] \arrow[ur, phantom, "\checkmark"] & P'(p) \arrow[u, "\mathrm{Colim}(P')_p"']
\end{tikzcd}$$

where $\mathrm{Colim}\,(P)_p$ and $\mathrm{Colim}\,(P')_p$ are the arrows of the colimit cocones $P \to cP$ and $P' \to cP'$.

Definition 3.3 (Category of systems)**.** Systems and morphisms of systems make a category $\boldsymbol{Sys}$.

Note that, due to the restriction of systems to the set V_{ω_1}, the category $\boldsymbol{Sys}$ is a small category.

Of course, systems are more interesting when they evolve through time. We extend the notion of evolutive systems [9] by adding the layers:

Definition 3.4 (Layered evolutive system)**.** A *layered evolutive system* (or LES) $\mathbb{L}$ is a (small) functor $\mathbb{L} : (T, \leqslant) \to \boldsymbol{Sys}$ such that $T \subset \mathbb{R}$ is a subset of real numbers, seen as a total order. A layered evolutive transformation $\mathbb{L} \to \mathbb{L}'$ is a usual natural transformation between the two functors $\mathbb{L}$ and $\mathbb{L}'$.

Layered evolutive systems and layered evolutive transformations form a category $\boldsymbol{LES}$.

A LES can be seen as a collection of snapshots of a given system evolving through time.

3.2 Link to Ehresmann and Vanbremeersch's work

Our construction is to be seen as an extension, and not as a replacement to [9]. We provide a few results that show how our extension relates to the main theory.

Proposition 3.5. *There is a forgetful functor* $\boldsymbol{Sys} \to \boldsymbol{Cat} : (\mathscr{C}, P) \to \mathscr{C}$*. It induces a forgetful functor* $\boldsymbol{LES} \to \boldsymbol{ES}$*.*

We distinguish two free functors:

Definition 3.6 (Layering functors)**.** The *trivial layering functor* is defined as:

$$\begin{cases} \boldsymbol{Cat} & \longrightarrow & \boldsymbol{Sys} \\ \mathscr{C} & \longmapsto & (\mathscr{C}, \Delta^1_{\boldsymbol{Cat}} 0 \mapsto \mathscr{C}) \end{cases}$$

Using some technicity, we could also define a *complete layering functor* which decomposes a category into a unique superposition of preorders. However, we do not need this functor in this paper, so we will not describe the technical details there.

Both functors are free functors, however they do not seem to be part of an adjunction with the forgetful functor $\boldsymbol{Sys} \to \boldsymbol{Cat}$.

Our main point was to give a translation of the MP (Definition 2.9) that correspond better to the concept of degeneracy. We need to adapt the notion of diagram.

Proposition 3.7. *Let $(\mathscr{C}, P : \mathscr{P} \to \boldsymbol{Cat}) \in \boldsymbol{Sys}$, and let $D : \mathscr{D} \to \mathscr{C}$ be a diagram.*

There exists a unique system $\left(\mathscr{D}, D^P\right)$ and a unique natural transformation $\delta^P = \left(\delta^P_p : D^P(p) \to P(p)\right)_{p \in \mathscr{P}}$ such that, for all $p \in \mathscr{P}$, the following diagram commutes:

$$\begin{tikzcd}
D^P(p) \arrow[r, "\delta^P_p"] \arrow[d, hook] \arrow[dr, phantom, "\checkmark"] & P(p) \arrow[d, hook, "\iota_p"] \\
\mathscr{D} \arrow[r, "D"'] & \mathscr{C}
\end{tikzcd}$$

Proof. Each δ^P_p is constructed as the pullback of D and ι_p. In detail, $D^P(p)$ is the subcategory of $\mathscr{D}$ with the same objects, with only the arrows of $\mathscr{D}$ that are the antecedents of the arrows of $\mathscr{C}$ that are also in $P(p)$.

□

Definition 3.8 (Layered diagram)**.** Let $\mathscr{S} = (\mathscr{C}, P : \mathscr{P} \to \boldsymbol{Cat}) \in \boldsymbol{Sys}$.

If $D : \mathscr{D} \to \mathscr{C}$ is a diagram, then we call *layered diagram* the natural transformation δ^P defined in Proposition 3.7.

We can now define:

Definition 3.9 (Layered Multiplicity Principle)**.** Let $\mathscr{S} = (\mathscr{C}, P : \mathscr{P} \to \boldsymbol{Cat})$ be a system. Consider two diagrams $D : \mathscr{D} \to \mathscr{C}$ and $E : \mathscr{E} \to \mathscr{C}$, as well as δ^P and ϵ^P, their respective layered diagrams.

We say that D et E verify the *Layered Multiplicity Principle* when there exists a $p_0 \in \mathscr{P}$ such that $\delta^P_{p_0}$ and $\epsilon^P_{p_0}$ verify the (usual) MP in $P(p_0)$ (Definition 2.9).

The idea is as follows. We have degeneracy when a system shows two subsystems, that have their own functionalities, and share one in particular. In a layered system, each functionality is depicted in a given layer. Thus, if two subsystems share a functionality, then the related layer should show the usual MP between these subsystems. Then, the superposition of layers should "hide" this MP by adding the other functionalities of the two subsystems. Hence Definition 3.9.

Note that the layered MP gives no condition on the functionality to be preserved. Maybe this functionality is best described in a layer that is completely different to the one showing the MP.

3.3 Resilience in categories

Recall that we see resilience as a set of three capacities: absorptivity, adaptivity and recoverability (see Section 2.2 for more information).

This definition calls for a partitioning of the system, with some parts dedicated to the capacities of resilience. Thus, we need to endow a system (as defined in Definition 3.1) with more structure.

Definition 3.10 (Divided system)**.** A *divided system* is a tuple

$$(\mathscr{C}, P, \mathrm{Atk}, \mathrm{Coreg}, \mathrm{Rep}, \mathrm{Dev}, \mathrm{Res}, \mathrm{Oth}, p_{\mathrm{Atk}}, p_{\mathrm{Coreg}}, p_{\mathrm{Rep}})$$

where:

1. $(\mathscr{C}, P)$ is a system
2. Dev, Res and Oth are disjoint subcategories of $\mathscr{C}$
3. Atk, Coreg and Rep are disjoints subsets of objects of $\mathscr{C}$
4. For all $\mathscr{X} \in \{\mathrm{Dev}, \mathrm{Res}, \mathrm{Oth}\}$, for all $x \in \{\mathrm{Atk}, \mathrm{Coreg}, \mathrm{Rep}\}$, $x \notin \mathscr{X}$
5. $\mathrm{Ob}(\mathrm{Dev} + \mathrm{Res} + \mathrm{Oth}) + \{\mathrm{Atk}, \mathrm{Coreg}, \mathrm{Rep}\} = \mathrm{Ob}(\mathscr{C})$
6. $p_{\mathrm{Atk}}, p_{\mathrm{Coreg}}, p_{\mathrm{Rep}} \in \mathscr{P}$
7. p_{Atk} (respectively, p_{Coreg}, p_{Rep}) are such that the only non-identity arrows of $P(p_{\mathrm{Atk}})$ (respectively, $P(p_{\mathrm{Coreg}})$, $P(p_{\mathrm{Rep}})$) have domain or codomain Atk, (respectively,Coreg, Rep)

A divided system is a setup for the study of resilience. We hope that the several divisions will be enough to represent resilience in the broadest and most meaningful sense, that of [25] (seen in Section 2.2).

The disjoint subcategories Dev, Res and Oth represent different parts of the system with different roles:

1. Dev is a subcategory representing the main device of the system
2. Res is a subcategory representing the resources available to the device, but not actually part of the main device
3. Oth represents the rest of the model that interacts with the main device

A typical description of a system follows.

The layer Atk represents the action of a perturbation on the system. Such a layer will typically "delete" one or more object(s) or arrow(s). This perturbation is what the system tries to respond to with resilience.

Firstly, the system tries to absorb this perturbation, and reduce its impact on the system performance. This is achieved by the Multiplicity Principle (MP) in strategic points. Roughly, the perturbation affects a subsystem verifying the MP, meaning roughly that it is replicated. Note that the MP is not represented in a distinguished layer because it is not supposed to be maintained due to the perturbation, which would cause modelling issues.

Secondly, the perturbation is over, and the system is damaged. The strategic points with multiplicity are damaged, which trigger an alert to the coregulator, represented in the layer Coreg. Its role is to adapt the internal structure of the system in order to maintain the functionalities formerly ensured by a multiplicity of subsystems.

Thirdly, the system has survived the perturbation and has adapted to its damage, but it is time to repair it. The reparation part is handled by the repairer, represented in the layer Rep. It uses the resources that are available to the system.

In this model we do not assume human intervention, but human action could be modelled by the strategies used by the coregulator and the repairer.

This will be made clearer in the following examples.

We are again interested in describing the evolution in time of such systems, as resilience is not instantaneous:

Definition 3.11 (Morphism of divided systems)**.** Let

$$\mathscr{D} = (\mathscr{C}, P, \mathrm{Atk}, \mathrm{Coreg}, \mathrm{Rep}, \mathrm{Dev}, \mathrm{Res}, \mathrm{Oth}, p_{\mathrm{Atk}}, p_{\mathrm{Coreg}}, p_{\mathrm{Rep}})$$

and

$$\mathscr{D}' = \left(\mathscr{C}', P', \mathrm{Atk}', \mathrm{Coreg}', \mathrm{Rep}', \mathrm{Dev}', \mathrm{Res}', \mathrm{Oth}', p'_{\mathrm{Atk}}, p'_{\mathrm{Coreg}}, p'_{\mathrm{Rep}}\right)$$

be two divided systems.

A *morphism of divided systems* $D : \mathscr{D} \to \mathscr{D}'$ is a tuple $(S, \sigma, D_{\mathrm{Atk}}, D_{\mathrm{Coreg}}, D_{\mathrm{Rep}})$ such that:

1. $(S, \sigma) : (\mathscr{C}, P) \to (\mathscr{C}', P')$ is a morphism of systems
2. For all $(\mathscr{X}, \mathscr{X}') \in \left\{(\mathrm{Dev}, \mathrm{Dev}'), (\mathrm{Res}, \mathrm{Res}'), \left(\mathrm{Oth}, \mathrm{Oth}'\right)\right\}$, the following diagram commutes:

$$\begin{tikzcd}
\mathscr{X} \arrow[r, "D_{\mathscr{X}}"] \arrow[d, hook] \arrow[dr, phantom, "\checkmark"] & \mathscr{X}' \arrow[d, hook] \\
\mathscr{C} \arrow[r, "D"'] & \mathscr{C}'
\end{tikzcd}$$

3. For all $(a, a') \in \left\{\left(\mathrm{Atk}, \mathrm{Atk}'\right), (\mathrm{Coreg}, \mathrm{Coreg}'), (\mathrm{Rep}, \mathrm{Rep}')\right\}$, $S(a) = a'$ and $p_a = p'_{a'}$

The take-home message of this definition, is that a morphism $D : \mathscr{D} \to \mathscr{D}'$ of divided systems must preserve the semantics of each subdivision of $\mathscr{D}$. For example, an object of the main device (represented by Dev) cannot become a resource in some layer $P(p)$ and an attacker in another layer.

Of course, we can derive a category:

Definition 3.12 (Category of divided systems)**.** Divided systems and morphisms of divided systems, together with the obvious component-wise composition, make a category ***DivSys***.

We argue that:

Hypothesis 3.13. Let

$$\mathscr{D} = (\mathscr{C}, P, \mathrm{Atk}, \mathrm{Coreg}, \mathrm{Rep}, \mathrm{Dev}, \mathrm{Res}, \mathrm{Oth}, p_{\mathrm{Atk}}, p_{\mathrm{Coreg}}, p_{\mathrm{Rep}})$$

be a divided system.

If the MP holds in some layer $P(p)$ of $\mathscr{D}$ within the subcategory Dev, meaning that the MP holds two diagrams D and E whose image is contained in Dev, then the represented system has some form of resilience.

This hypothesis makes the connection between resilience and the (layered) MP (Definition 3.9). We illustrate this hypothesis with the following example.

4 An application to drones

We give an application to illustrate the modelling introduced in Section 3.

This example is thought as a simplification of a realistic situation, although it is still polished in order to be simulated. This example is loosely inspired by the drone swarm described in [27]. However, it is abstracted enough so that it can still be generic. With little adaptation, this study could apply to many multi-agent systems.

We first describe the system, we give a system-dependent definition of resilience, then we describe the model following the formalism of Definition 3.10, and where its resilience lies. We conclude with a simulation to illustrate our point.

4.1 Description of the system

The system consists of N aerial drones scanning for weed in a field of potatoes. For simplicity of simulation, we assume that the field consists of a square of $n_c \times n_c$ of cells. Each cell c has a density ρ_c of weed that the drone swarm tries to estimate.

The common program of the drones is simple:

1. Each drone D visits one cell c at a time
2. It scans for weed; it takes a certain time τ_{scan}
3. It sends its i-th estimation $\widehat{\rho}_{D,c,i}$ of weed density in the cell c to all the other drones
4. It finds a new cell to scan
5. It moves to the new cell with a certain speed s
6. Repeat until each cell c has been scanned T times
7. The final estimation of weed density of each cell c is the average of all the estimations of all drones in the swarm for that cell c.

We assume that all drones are connected together and that the transfer of information is immediate (or at least, negligible compared to the movements of the drones and the scanning time τ_{scan}) and perfect (there is no communication error).

All drones store the same array $\widehat{\rho}$ indexing the estimations of weed density per cell. When one drone D makes its i-th estimation $\widehat{\rho}_{D,c,i}$ for a cell c, it appends it to the array $\widehat{\rho}$. Thus, the whole drone swarm knows every estimation at each instant.

This simulation is not about the detection algorithm of drones. For each cell, the drone guesses a random Gaussian variable centered on the actual weed density. Thus, the higher the number of scans, the better the estimation, as it will be computed as the mean of all estimations.

Once a drone is done estimating the weed density in the current cell, it will find another cell. The drone will consider all cells around it:

1. If a cell is being scanned by another drone, or another drone is in transit to this cell, the cell is ignored
2. The further the cell, the lower the chance to visit it
3. If a cell has already been scanned T times, it is ignored

Drones have a parameter that controls the "radius" of cells they consider, we denote this by r. The probability of choosing cell c when in position c_{old} is a discrete Gaussian proportional to:

$$P\left[c|c_{\text{old}}\right] \propto e^{\frac{\|c-c_{\text{old}}\|_2^2}{2r^2}}$$

We compute this value for each cell c. Cells c that should be ignored have probability 0 and we normalise the computed values. We then run a roulette-wheel selection to choose the next visit.

The drones scan the field repeatedly until all the cells have been scanned T times.

This is a slightly modified version of the drone swarm described in [27]. In the following, we extend this model to include a threat, in order to study how a coregulator and a repairer could be added to endow this swarm with resilience.

The threat will be an instant, generic attacker or malfunction that put one drone out of service in one attack. The attacker repeat the attack several times until either all the drones are destroyed, or the swarm is done scanning the field.

When a drone is attacked, it stops working. We assume that the attack takes a time τ_a and, the system does not have the means to prevent this attack.

For our purposes, we will add a few more elements: a reserve of drones, a coregulator, and a repairer. The reserve consists in a number N_R of more drones that are ready to join the swarm of drones just in case. The coregulator and repairer are two other agents whose roles will be to orchestrate the swarm and reserve. Further details about these agents will be discussed in Section 4.3.

4.2 Definition of resilience

We denote by $\mathcal{S}$ this swarm.

What does it mean for $\mathcal{S}$ to be resilient?

Resilience is the ability to maintain a certain level of performance even under disturbance (Section 2.2). In our case, the disturbance is this instant, generic threat that attacks a drone periodically.

We can measure the performance of the drones:

1. the difference between their estimation of the density and the real value
2. the time the drones take to reach a certain level of precision

Thus, in this example, resilience of $\mathcal{S}$ boils down to completing the scan of the field to a certain extent (say, a certain level of error), despite the attack. Following the definition of resilience in Section 2.2, our swarm of drones should have three capacities:

1. An *absorptive capacity*: here, the attacker will only take down the drones one by one, so the absorptive capacity boils down to having more than one drone in the swarm.
2. An *adaptive capacity*: the rest of the swarm passively notices that one drone has been taken down; this reduces the number of drones scanning the field, thus each drone knows that it must scan more cells. This is a passive adaptation because the drones will only stop scanning the field after each cell has been scanned a certain number of times. If there are less drones, then this number will be attained more slowly.
3. And a *recoverability*: the coregulator and repairer work together to dispatch drones from the reserve to replace the missing drone and try to maintain the work of the swarm. This is an obvious example of recovery.

We make the choice to only work on the structure of the swarm. We assume that most parameters of the drones are fixed by the provider, and that we cannot alter them.

According to Hypothesis 3.13, there should be resilience in the swarm. So, we should have a layer that decomposes the swarm into drones, and the MP should be visible in this layer.

4.3 Description of the model

In this section, we describe the categorical model

$$(\mathscr{C}, P, \mathrm{Atk}, \mathrm{Coreg}, \mathrm{Rep}, \mathrm{Dev}, \mathrm{Res}, \mathrm{Oth}, p_{\mathrm{Atk}}, p_{\mathrm{Coreg}}, p_{\mathrm{Rep}})$$

of system consisting of the drone swarm, the field, and the rest. It is easier to describe each layer, then superpose them. So, in this section, we describe each layer, step by step. We also discuss their evolution in time and the superposition of different layers.

We first discuss the layers, and then we discuss the resilience of the system.

General structure, and layer of objects. We start by describing the general structure of this system. To do so, we decide to first describe the most basic layer, that of objects. The layer of objects consists of six sets of objects:

1. A set of drones $(D_i)_{i \in N}$
2. An object S representing the swarm
3. A set of cells $(c_{i,j})_{i,j \in n_c}$
4. An object F representing the field
5. An object A representing the attacker or generic threat (thus $\mathrm{Atk} = \{A\}$)
6. An object C representing the coregulator (thus $\mathrm{Coreg} = \{C\}$)
7. An object R representing the repairer (thus $\mathrm{Rep} = \{R\}$)

However, drones cannot reproduce. Without anything more, we cannot achieve the third capacity of resilience, restoration, because as such, the system cannot repair itself. We would need more drones. Thus, in addition to the objects described above, we add new elements to the system in the form of a reserve $(R_l)_{l \in N_R}$ of drones. These drones wait for the call of the repairer to join the swarm when needed.

The layer of objects is thus:

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 & \dots & \dots & D_k & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} & \dots & D_N & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd} \tag{drones-objects}$$

Using the formalism of Definition 3.10, we have:

1. The main device Dev is the subcategory consisting of the object S for the swarm and the objects D_k for the drones
2. The resources Res is the subcategory consisting of the objects R_l representing the backup drones
3. The rest of the category forms subcategory Oth, which includes the object F for the field and $C_{i,j}$ for the cells

To complete the description of this system as a divided system, we need to describe the decomposition in layers $P : \mathscr{P} \to \mathscr{C}$ and a few specific layers. The description of $P : \mathscr{P} \to \mathscr{C}$ is the purpose of this section, and p_{Atk}, p_{Coreg} and p_{Rep} will be described in their own paragraphs.

Defining these divisions allows to ensure that the semantics of the objects will remain the same throughout the description of the layers; i.e. if D_i represents drone i in (drones-objects), then it also represents the same drone in all other layers.

The evolution in time of this layer is as follows:

1. The layer loses drones depending on the action of the attacker A

2. The layer "transfers" drones from Res to Dev depending on the action of the repairer (we will see later what we mean)
3. We assume that the field is left unchanged, as natural disasters are not part of the model

Layer of decomposition. Of course, the drones compose the swarm, and the cells compose the field. It defines a new layer:

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 \arrow[urr] & \dots & \dots & D_k \arrow[ul] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ul] & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] & \dots & D_N \arrow[uuul] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

(drones-decomp)

The idea is that the swarm can be studied as a single object, and if we zoom on it, we can decompose the swarm into its drones. Note that this is the intent behind the use of the colimits in the model described in [9].

In this layer, the swarm S (respectively the field F) is de facto the coproduct of the set of drones (resp. of the set of cells).

The swarm and the field are respectively the coproducts of the drones and the cells in the layer (drones-decomp). Should the coproduct property be preserved in the final layer?

This is to the modeller to decide. However, this goes with the "spirit" of the model. In the whole system, the swarm is still supposed to decompose into its several drones, and the field is still supposed to decompose into its several cells.

The evolution in time is similar to the layer of objects:

1. The layer loses drones depending on the action of the attacker A, and we remove the corresponding arrow when that happens.
2. When the repairer "transfers" drones from Res to Dev (more information on this later), we add an arrow from that drone R_l to the swarm S

In this example, drones are either scanning a cell, moving to a cell, or under attack. Each of these states need a layer.

Layer of scan. When a drone is scanning a cell, we draw an arrow between the drone and the corresponding cell in the layer of scanning. Not all drones scan all the time, so some drones will not have an arrow in this layer.

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 & \dots & \dots & D_k \arrow[ddrrr] & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uurrr, bend right] & \dots & D_N & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd} \tag{drones-scan}$$

As for its evolution in time, this layer may lose drones depending on the action of the attacker, and a drone could move from the reserve of drones to the swarm. Also, the drones may gain or lose an arrow depending on whether they start or are done scanning the cell.

Layer of movement. When a drone is moving to a cell, we draw an arrow between the drone and the corresponding cell. Again, not all drones are moving all the time, so not all drones have arrows in this layer.

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 \arrow[rrrrrr, bend left] & \dots & \dots & D_k & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} & \dots & D_N \arrow[r] & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd} \tag{drones-mvmt}$$

Similarly to the layer of scanning, this layer may lose drones depending on the action of the attacker, and a drone could move from the reserve of drones to the swarm depending on the action of the repairer. Also, arrows may appear or disappear depending on whether a drone starts or is done scanning a cell.

Layer of the attacker. To complete the description using the formalism of Definition 3.10, we now have to describe the action of the attacker, the coregulator and the repairer. We start with the attacker.

Recall that the attacker is a generic, instant threat to the swarm. We draw one arrow between the attacker and each drone it is attacking; when it is done with this target, the drone disappears, and a new arrow is drawn between the attacker and another drone.

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 & \dots & \dots & D_k & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} & \dots & D_N & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A \arrow[uur] \arrow[ur] \arrow[urr] & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd} \tag{drones-atk}$$

Following Definition 3.10, this layer is exactly $P\left(p_{\text{Atk}}\right)$.
The evolution in time of this layer is a bit different than the other layers:

1. Most of the time, this layer has no non-identity arrow at all
2. At a given instant, and for a small interval of time, there is one arrow $A \to D_k$ for each attacked D_k
3. After the attack, new arrows $D_k \to A$ appear, meaning that the drones are now under the control of the attacker

Layer of the coregulator. The coregulator is assumed to work on all drones at once (changing the speed, Gaussian spread and time of detection). We thus need an arrow from the coregulator R to every drone D_k.

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 & \dots & \dots & D_k & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} & \dots & D_N & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A & & C \arrow[uuull, bend left] \arrow[ul] \arrow[uu] \arrow[uuur, bend right] \arrow[ur] & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd} \tag{drones-coreg}$$

Following Definition 3.10, this layer is exactly $P\left(p_{\text{Coreg}}\right)$.
The evolution in time is similar to that of the attacker:

1. In the general case, there is no arrow in this layer
2. After the attack, the coregulator detects the missing drones and draws an arrow from C to each remaining drone D_k corresponding to a given order to the drone
3. Once the drones have obeyed, the arrows are removed from the layer

Layer of the repairer. Finally, the last layer is that of the repairer. As drones are not cells, we cannot assume that a drone can reproduce (unless this is a self-replicating von Neumann automaton), so we choose to represent the reparation of the swarm by using a reserve of drones. The role of the repairer is thus simple: transfer the drones from the reserve to the swarm. This is easy to represent: we draw an arrow from the repairer R to the chosen drone R_l:

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 & \dots & \dots & D_k & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} & \dots & D_N & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R \arrow[ur] & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd} \tag{drones-rep}$$

Following Definition 3.10, this layer is exactly $P\left(p_{\text{Rep}}\right)$.
The evolution in time of this layer is as follows:

1. Before the attack, during the attack and for a certain amount of time after the attack, this layer has no non-identity arrows
2. After the coregulator has given new orders to the remaining drones (after a certain time given by the model), the repairer counts the number of missing drones in the swarm, and we add an arrow $R \to R_l$ for each chosen drone R_l from the reserve
3. After a certain amount of time (again decided by the model), the chosen drone integrates the swarm. This is represented by the addition of a new arrow $R_l \to S$ in the layer (drones-decomp) as shown below:

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 \arrow[urr] & \dots & \dots & D_k \arrow[ul] & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] & \dots & D_N \arrow[uuul] & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A & & C & & R_1 \arrow[uuuull] & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

Layer of action. So far, the diagram P looks like this:

$$\begin{tikzcd}
 & \text{(drones-decomp)} \\
 & \text{(drones-scan)} \\
\text{(drones-objects)} \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] \arrow[dddr, bend right] & \text{(drones-mvmt)} \\
 & \text{(drones-atk)} \\
 & \text{(drones-coreg)} \\
 & \text{(drones-rep)}
\end{tikzcd}$$

If the system is not under attack, that is, if (drones-atk), (drones-coreg) and (drones-rep) consist only of objects without arrows, the system is entirely described by the layers (drones-decomp), (drones-scan) and (drones-mvmt). Their superposition gives:

$$\begin{tikzcd}
 & & S & & & F & & \\
D_1 \arrow[urr] \arrow[rrrrrr, bend left] & \dots & \dots & D_k \arrow[ul] \arrow[ddrrr] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ul] & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] \arrow[uurrr, bend right] & \dots & D_N \arrow[uuul] \arrow[r] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

Seeing that each drone (being part of the swarm) has some action on one of the cells (being part of the field), it becomes natural to expect an arrow from S (representing the swarm) to F (representing the field). We cannot simply add a new layer containing only an arrow $S \to F$, because, when superposing the layers, we cannot expect all diagrams to commute correctly, i.e. computing the colimit does not yield $D_k \to S \to F$ to be equal to $D_k \to C_{i,j} \to F$. One solution is to add another layer (drones-action), that contains the superposition of (drones-decomp), (drones-scan) and (drones-mvmt), and that defines the new arrow $S \to F$ such that $D_k \to S \to F$ is equal to $D_k \to C_{i,j} \to F$.

$$\begin{tikzcd}
 & & S \arrow[rrr] & & & F & & \\
D_1 \arrow[urr] \arrow[rrrrrr, bend left=10] & \dots & \dots & D_k \arrow[ul] \arrow[ddrrr] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ul] & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] \arrow[uurrr, bend right] & \dots & D_N \arrow[uuul] \arrow[r] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd} \tag{drones-action}$$

Note that for each drone D_k scanning or moving to a given cell $C_{i,j}$, the following diagram commutes:

$$\begin{tikzcd}
S \arrow[r] & F \\
D_k \arrow[u] \arrow[r] & C_{i,j} \arrow[u]
\end{tikzcd}$$

This ensures the equality of the compositions $D_k \to S \to F$ and $D_k \to C_{i,j} \to F$. It can be interpreted as follows. The contribution of drone D_k to the swarm S acting on the field F is exactly the fact that D_k scans or is travelling to scan $C_{i,j}$, which is part of the field. The composition law conveys a form of transitivity.

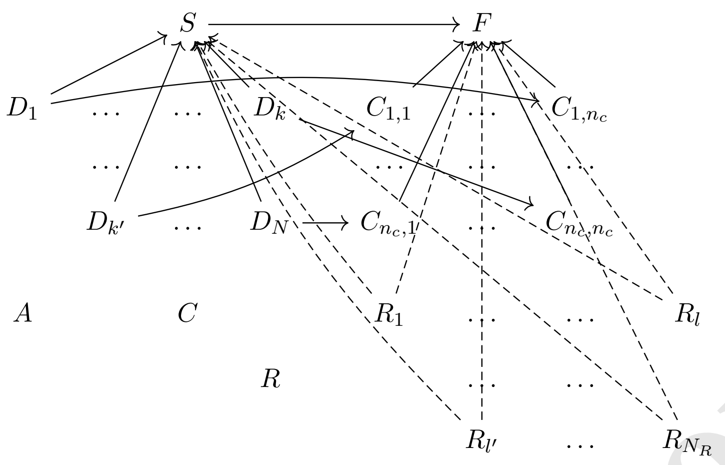
Layer of reserve. Our final layer represents the utility of the reserve. The drones in the reserve are meant to join the swarm whenever the repairer sees fit. Its arrows are dashed to emphasize that they are more about a potential activity, rather than an actual activity.

$$\begin{tikzcd}
 & & S \arrow[rrr] & & & F & & \\
D_1 & \dots & \dots & D_k & C_{1,1} & \dots & C_{1,n_c} & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} & \dots & D_N & C_{n_c,1} & \dots & C_{n_c,n_c} & \\
A & & C & & R_1 \arrow[uuuull, dashed] \arrow[uuuur, dashed] & \dots & \dots & R_l \arrow[uuuulllll, dashed] \arrow[uuuulll, dashed] \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} \arrow[uuuuuulll, dashed] \arrow[uuuuuu, dashed] & \dots & R_{N'_R} \arrow[uuuuuulllll, dashed] \arrow[uuuuuull, dashed]
\end{tikzcd} \tag{drones-reserve}$$

Each drone R_l in the reserve has an arrow $R_l \to S$ to emphasize that they can join the swarm, and an arrow $R_l \to F$ to emphasize that they may work on the field even if it has not been assigned a cell yet.

The evolution in time of (drones-reserve) is simple as well: if the repairer asks the drone represented by R_l to join the swarm, it loses its arrows in (drones-reserve).

By superposing (drones-action) and (drones-reserve), we obtain a combined layer that we call (drones-potential):



(drones-potential)

For the sake of readability, we do not represent the dashed arrows in the following sections. However, this layer will be very useful for the study of the MP.

Superposition of layers. Now considering the layers (drones-action), (drones-reserve) and (drones-potential) to P, we obtain:

$$\begin{tikzcd}
 & \text{(drones-reserve)} \arrow[rr, bend left] & & \text{(drones-potential)} \\
 & \text{(drones-decomp)} \arrow[r] & \text{(drones-action)} \arrow[ur] & \\
 & \text{(drones-scan)} \arrow[ur] & & \\
\text{(drones-objects)} \arrow[uuur, bend left] \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] \arrow[dddr, bend right] & \text{(drones-mvmt)} \arrow[uur] & & \\
 & \text{(drones-atk)} & & \\
 & \text{(drones-coreg)} & & \\
 & \text{(drones-rep)} & &
\end{tikzcd} \tag{1}$$

The typical chain of events is as follows. First, the swarm of drones performs its task normally. At some point in time, the swarm is damaged by a perturbation, the swarm loses drones. The number of drones ensures that the perturbation does not decimate the whole swarm. The coregulator detects the missing drones, and sends commands to them changing some internal parameters (movement speed, scanning speed), to temporarily compensate for the loss of drones, and save some time (for example, if time is a constraint). At the same time, the repairer is aware of the loss of drones, and chooses a few substitute from the reserve. Naturally, we assume that the reserve is physically far from the field, so it takes time for the drones to physically join the swarm.

Following this typical scenario, all the layers described above superpose to give different versions of $\operatorname{Colim}(P)$ (depending on the instant taken into account). For the sake of readability, we do not represent all compositions; we assume that this is a graph generating a category.

1. In the normal case, the only layers with non-identity arrows are (drones-decomp), (drones-mvmt), (drones-scan) and (drones-reserve). Their superposition is included in the layer (drones-potential) described above[1].
2. When the attack happens, (drones-atk) gets arrows $A \to D_k$ for some D_k's:

$$\begin{tikzcd}
 & & S \arrow[rrr] & & & F & & \\
D_1 \arrow[urr] \arrow[rrrrrr, bend left] & \dots & \dots & D_k \arrow[ul] \arrow[ddrrr] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ul] & \\
 & \dots & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] \arrow[uurrr, bend right] & \dots & D_N \arrow[uuul] \arrow[r] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A \arrow[uur] \arrow[ur] \arrow[urr] & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

Note that, by composition of arrows, the attacker also attacks S whenever it attacks at least one drone.
3. After the attack, the swarm loses some drones, but they don't just disappear from the model. We can represent this loss by new arrows from the lost drones to the attacker, meaning that they are now tagged as "attacked" and no longer belong to the swarm:

$$\begin{tikzcd}
 & & S \arrow[rrr] & & & F & & \\
D_1 \arrow[urr] \arrow[rrrrrr, bend left] & \dots & \dots & D_k \arrow[ul] \arrow[ddrrr] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ul] & \\
 & \dots \arrow[ddl] & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] \arrow[uurrr, bend right] \arrow[dl] & \dots \arrow[dll] & D_N \arrow[uuul] \arrow[r] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

4. Then the coregulator detects that some drones are missing and applies some changes to the behaviour of the drones:

[1]Remember that, for the sake of readability, we do not represent here the dashed arrows of (drones-reserve).

$$\begin{tikzcd}
 & & S \arrow[rrr] & & & F & & \\
D_1 \arrow[urr] \arrow[rrrrrr, bend left] & \dots & \dots & D_k \arrow[ul] \arrow[ddrrr] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ull] & \\
 & \dots \arrow[dddl] & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] \arrow[uurrr, bend right] \arrow[dl] & \dots \arrow[dll] & D_N \arrow[uuul] \arrow[r] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A & & C \arrow[uuull, bend left] \arrow[u] \arrow[ur] \arrow[uuur, bend right] & & R_1 & \dots & \dots & R_l \\
 & & & R & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

5. Finally, the repairer designates a few drones to replace the missing ones:

$$\begin{tikzcd}
 & & S \arrow[rrr] & & & F & & \\
D_1 \arrow[urr] \arrow[rrrrrr, bend left] & \dots & \dots & D_k \arrow[ul] \arrow[ddrrr] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ull] & \\
 & \dots \arrow[dddl] & \dots & & \dots & \dots & \dots & \\
 & D_{k'} \arrow[uuur] \arrow[uurrr, bend right] \arrow[dl] & \dots \arrow[dll] & D_N \arrow[uuul] \arrow[r] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A & & C & & R_1 & \dots & \dots & R_l \\
 & & & R \arrow[ur] \arrow[rr] & & \dots & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

and after some time, the new drones join the swarm:

$$\begin{tikzcd}
 & & S \arrow[rrr] & & & F & & \\
D_1 \arrow[urr] \arrow[rrrrrr, bend left] & \dots & \dots & D_k \arrow[ul] \arrow[ddrrr] & C_{1,1} \arrow[ur] & \dots & C_{1,n_c} \arrow[ull] & \\
 & \dots \arrow[dddl] & \dots & & \dots & \dots & \dots & \\
R_1 \arrow[uuurr] & D_{k'} \arrow[uuur] \arrow[dl] & \dots \arrow[dll] & D_N \arrow[uuul] \arrow[r] & C_{n_c,1} \arrow[uuur] & \dots & C_{n_c,n_c} \arrow[uuul] & \\
A & & C & & & R_3 & \dots & R_l \\
 & & & R & & & \dots & \\
 & & & & & R_{l'} & \dots & R_{N_R}
\end{tikzcd}$$

Note that all items of Definition 3.10 were defined throughout this section. In summary:

1. $\mathscr{C}$ is the superposition of all layers

2. $P : \mathscr{P} \to \boldsymbol{Cat}$ has domain $\mathscr{P}$ equal to the category generated by Diagram 1
3. $\mathrm{Dev} = (D_i)_{i \in N} \cup \{S\}$ across all layers
4. $\mathrm{Res} = (R_l)_{l \in N_R}$ across all layers
5. $\mathrm{Oth} = (c_{i,j})_{i,j \in n_c} \cup \{F\}$ across all layers
6. $\mathrm{Atk} = \{A\}$ across all layers
7. $\mathrm{Coreg} = \{C\}$ across all layers
8. $\mathrm{Rep} = \{R\}$ across all layers
9. $p_{\mathrm{Atk}} =$ (drones-atk) in Diagram 1, and $P(p_{\mathrm{Atk}})$ is the category drawn in (drones-atk)
10. $p_{\mathrm{Coreg}} =$ (drones-coreg) in Diagram 1, and $P(p_{\mathrm{Coreg}})$ is the category drawn in (drones-coreg)
11. $p_{\mathrm{Rep}} =$ (drones-rep) in Diagram 1, and $P(p_{\mathrm{Rep}})$ is the category drawn in (drones-rep)

Also note that each layer has objects $\mathrm{Dev} + \mathrm{Res} + \mathrm{Oth} + \mathrm{Atk} + \mathrm{Coreg} + \mathrm{Rep}$, and their semantics do not change across all the layers (even if they can while the system evolves, e.g. when a new drone comes to replace the attacked drones).

Denote by $\mathcal{S}$ the divided system gathering all this data. We thus have a divided system. We can now study its resilience.

Study of multiplicity. Our thesis is that if the Multiplicity Principle holds in a layer, then the represented system must be resilient (Definition 3.9). We argue that the resilience of the swarm is best described in (drones-potential). Is is easy to see that S is the colimit of the diagram consisting of all the drones $(D_k)_{k \in N}$. It is also the colimit of the diagram consisting of all the drones in the reserve $(R_l)_{l \in N_R}$. Due to the structure of the system, S is also the colimit of any non-empty subset of drones of the swarm or of the reserve.

Fact 4.1. *Let $(D_k)_{k \in I}$ be a proper subset of $(D_k)_{k \in N}$ in (drones-decomp).*

There is a unique cluster $(D_k)_{k \in I} \to (D_k)_{k \in N}$. Besides, there is no Multiplicity Principle between $(D_k)_{k \in I}$ and $(D_k)_{k \in N}$ in (drones-decomp).

Proof. This is a direct application of Proposition 2.10. □

In Fact 4.1, the MP does not need to hold between a diagram and a subdiagram, because as stated before, the MP is a sign that two subsystems are essentially independent, which we cannot assume of a subdiagram. Also note that this fact holds in all other layers that contain (drones-decomp).

Fact 4.2. *Let I be a subset of N and I_R be a non-empty subset of N_R.*

The MP holds between $D_R = (D_k)_{k \in I} \cup (R_l)_{l \in I_R}$ and $D_N = (D_k)_{k \in N}$ in (drones-potential).

Proof. D_R and D_N are different sets, and neither is a subset of the other. Thus, there is no cluster between either.

S is the coproduct of both D_N and D_R. Besides, $\boldsymbol{Cocones}(D_N) \cong \boldsymbol{Cocones}(D_R)$ (there is only one cocone, the one with peak S), but as there is no cluster between D_R and D_N, the isomorphism cannot be written as ΩG for some cluster G. Thus, the MP holds. □

Fact 4.3. *Let I and I' be subsets of N, and I_R and I'_R be non empty subsets of N_R, such that $I \cup I_R$ and $I' \cup I'_R$ are not included in one another.*

The MP holds between $D_R = (D_k)_{k \in I} \cup (R_l)_{l \in I_R}$ and $D'_R = (D_k)_{k \in I'} \cup (R_l)_{l \in I'_R}$ in (drones-potential).

Proof. The argument is very similar to that of Fact 4.2. □

Informally, Fact 4.1 states that the MP cannot be obtained from within the swarm of drones. We need a reserve. Fact 4.2 states that the MP holds between any (non-empty) subset of the current swarm, and any (non-empty) subset of the swarm to which we add some drones from the reserve. Similarly, Fact 4.3 states that the MP is maintained between any (non-empty) subset of drones and reserve, and any other (non-empty) subset of drones and reserve. Fact 4.2 justifies that the replacement of drones suffices for the MP to hold, and Fact 4.3 justifies that all subsequent replacements still maintain the MP.

These Facts support our design described in Section 4.3.

Fact 4.4. *The described system $\mathcal{S}$ verifies the layered Multiplicity Principle (Definition3.9).*

In (drones-potential), after the new drones have joined the swarm in step 5, the newly-constituted swarm (with the new drones but without the attacked ones) and the former swarm (with the attacked drones but without the new ones) share the same functionality, are relatively independent, and thus verify the MP. The system maintains the MP, as well as itself, as long as the reserve is non-empty. This supports Hypothesis 3.13.

4.4 Simulations

In this section, we describe a simulation to illustrate empirically the benefits of our approach.

Our simulation is loosely inspired by [27] and was described more thoroughly in Section 4.1. It is about a swarm of drones scanning a field of potatoes, looking for weed. Each drone scans each subdivision of the field (called cell). We add an attacker that will periodically take down one drone. Fortunately, we endow the swarm with a coregulator that will check the state of the swarm, and a repairer that will command to the drones in the reserve to join the swarm.

We summarise here the parameters:

1. T_{sim} is the simulation time (not the actual duration of the experiment)
2. N is the number of drones
3. $n_c \times n_c$ is the dimension of the field F
4. Each cell c is described by a position $(i, j) \in n_c \times n_c$ and has a weed density $\rho_c \sim \mathcal{U}(0, 1)$
5. All drones have the same:
   - 5.1. Starting position Start = $(x_{\text{start}}, y_{\text{start}})$ (which can be outside of the field)
   - 5.2. Speed s_{drone} to move from cell to cell
   - 5.3. Scanning time τ_{scan} to scan each cell
   - 5.4. Radius r of cells to consider for scanning
   - 5.5. Waiting duration τ_{wait} to attempt to find a new cell
   - 5.6. Waiting threshold τ_{thres} after which they decide that they are done
6. All drones will try to scan T times the field, but each cell will be scanned no more than $N \times T$ times
7. The attacker Atk takes time τ_a to attack a drone
8. The coregulator Coreg scans the swarm at a τ_{coreg} period
9. The repairer has no delay or other characteristic
10. N_{res} is the number of drones in the reserve
11. The reserve Res is at position $(x_{\text{res}}, y_{\text{res}})$ (which can be outside of the field)

The simulation is not meant to be a realistic representation of the work of a swarm of drones, nor is it a demonstration of drone synchronisation. Our goal is only to provide an argument illustrating our layered systems. For example, we do not consider extra distance between cells: the distance between cell (i_1, j_1) and (i_2, j_2) will simply be the l_2-distance

Table 1: List of values used in the simulations.

Parameter	Value
Simulation time	$T_{\text{sim}} = 2000$
Number of runs	$N_{\text{runs}} = 100$
Number of drones (swarm)	$N = 20$
Number of drones (reserve)	$N_{\text{res}} = 100$
Field size	$n_c \times n_c = 10 \times 10$
Starting position	Start = $(x_{\text{start}} = -5, y_{\text{start}} = -5)$
Reserve position	Res = $(x_{\text{res}} = 10, y_{\text{res}} = -5)$
Drone speed	$s_{\text{drone}} = 2$
Scanning time	$\tau_{\text{scan}} = 2.0$
Drone radius	$r = 3.0$
Drone waiting duration	$\tau_{\text{wait}} = 1.0$
Drone waiting threshold	$\tau_{\text{thres}} = 5.0$
Number of scan per drone per cell	$T = 10$
Attacker period	$\tau_a = 5, 10, 15, 20$
Coregulator period	$\tau_{\text{coreg}} = 50.0$

$\sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$. The same formula is used for the distance between cell (i, j), the starting position of the swarm $(x_{\text{start}}, y_{\text{start}})$ and the reserve $(x_{\text{res}}, y_{\text{res}})$.

The attacker must not be too aggressive, otherwise there is no resilience to show. First, the attacker must not decimate the whole swarm before the first scan has even happened:

$$\tau_a > \frac{\max\limits_{c \text{ cell}} (\|c - \text{Start}\|_2)}{s_{\text{drone}}} + \tau_{\text{scan}}$$

Similarly, the attacker must not take down drones from the reserve before its first scan has even happened:

$$\tau_a > \frac{\max\limits_{c \text{ cell}} (\|c - \text{Res}\|_2)}{s_{\text{drone}}} + \tau_{\text{scan}}$$

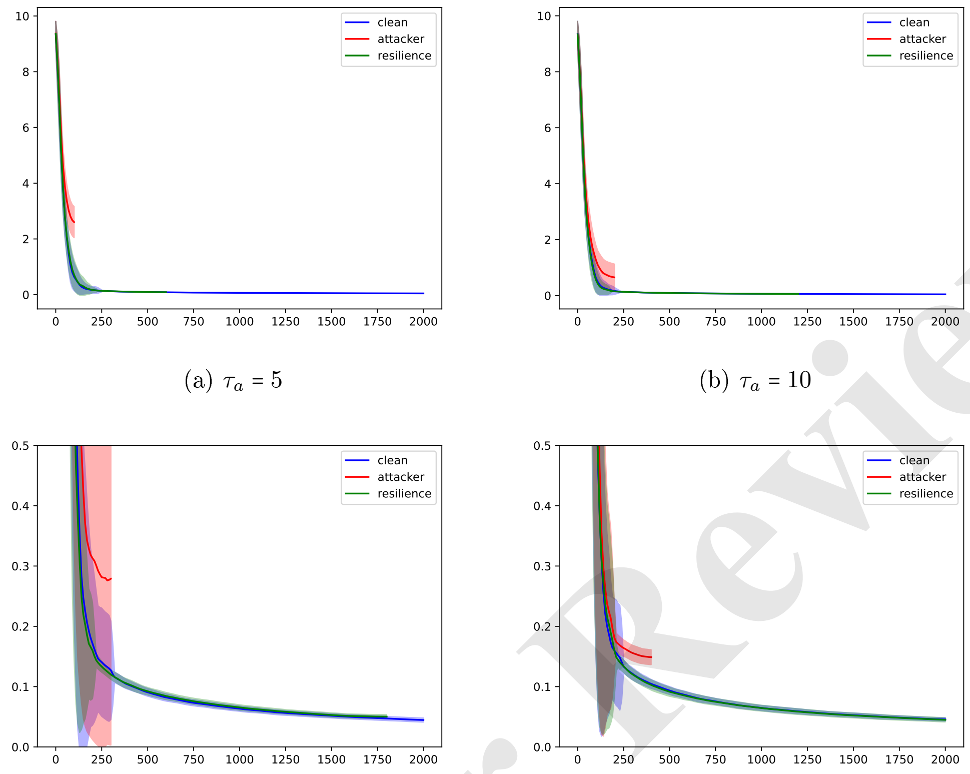
Also the attacker must be aggressive enough as to prevent the starting swarm (meaning, the swarm without drones coming from the reserve) to finish scanning the field. Otherwise, it's almost as if there was no attack. Roughly:

$$\tau_a < \frac{\max\limits_{c \text{ cell}} (\|c - \text{Start}\|_2)}{s_{\text{drone}}} + T \times \left(\frac{r}{s_{\text{drone}}} + \tau_{\text{scan}} \right)$$

Finally, we should expect the attacker to decimate the whole swarm in time $N \times \tau_a$, and the whole swarm plus reserve in time $(N + N_{\text{res}}) \times \tau_a$. If the drones are given a specific time T_{sim} to scan the whole field, then the swarm survives only if $N \times \tau_a > T_{\text{sim}}$, and the swarm plus reserve survive only if $(N + N_{\text{res}}) \times \tau_a > T_{\text{sim}}$. For these simulations, we assume that the swarm survives if at least one drone remains at the end of the simulation time T_{sim}.

The values used in this simulations are shown in Table 1.

Figure 1 displays the results of the simulation. The curves show the evolution of the loss (difference between the measured weed density, and the real density). The values are averaged over $N_{\text{runs}} = 100$ runs. The blue curve (called "clean") represents the simulations without attacker; this is the ideal case. The red curve (called "attacker") represents the simulations with an attacker but without the architecture of a divided system; this is the



(a) $\tau_a = 5$ (b) $\tau_a = 10$

(c) $\tau_a = 15$ (d) $\tau_a = 20$

Figure 1: Evolution of the estimation error with different levels of aggressivity of the attacker. In all cases, without resilience, the attacker decimates the swarm of drones before the end of the scan, and the resilient case follows closely the case without attacker.

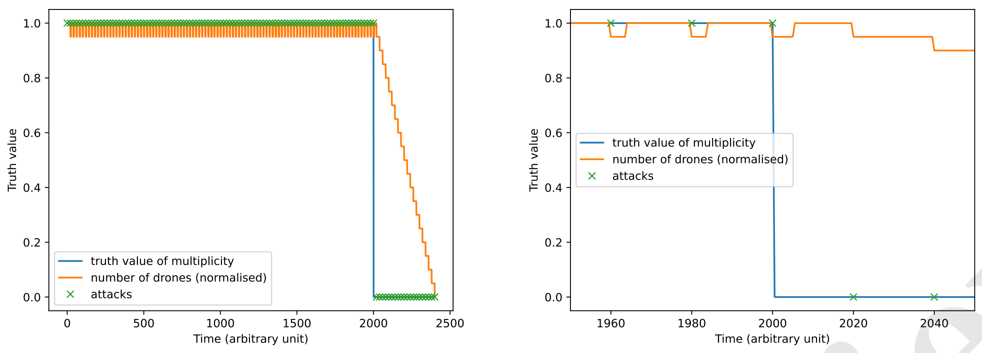
worst case. The green curve (called "resilience") represents the simulations with attacker but with the architecture of a divided system; this is a medium case.

The swarms implementing the divided system architecture show very similar losses to the swarms without attacks, as the green and blue curves are almost identical. This does not seem to depend on the aggressivity of the attacker, as we get similar results for each value of τ_a. However, as expected, the attacker still decimates the swarm and reserve if too aggressive, but this does not prevent the swarm from achieving results very similar to the swarm without attacker. We could call resilient the swarm with the divided system architecture.

Figure 2 shows the evolution in time for one run of the simulation. The MP holds as long as there is a drone in the reserve, and this corresponds to the interval of time when the system can still bounce back to the initial state. If the reserve is empty, every attack, every lost drone reduces the performance of the whole swarm, which goes in parallel with a loss of MP.

5 Conclusion

Resilience is a desirable property for critical systems. We extended the language of Memory Evolutive Systems proposed in [9] to improve the representability of this property in category theory. The definition of resilience by [25] appeared to be the most suitable due



(a) Global view. (b) Zoomed.

Figure 2: Evolution of the MP throughout the simulation for a given run. The left-hand figure shows the global evolution of the MP through time. The right-hand figure shows the evolution of the MP around the moment when the MP no longer holds.

to its generality for this broad definition of systems. We thus have a setting to discuss resilience.

Note that our extension is still compatible with all the theorems and results about MES, despite the added complexity and structure, as those results now apply on the layers instead of the complete configuration. This article thus describes a solution to the problems described in introduction (potentially meaningless compositions and partial translation of degeneracy to the MP), that smoothly inserts into Ehresmann and Vanbremeersch's theory.

From the example in Section 4, we see that it is easier to opt for a bottom-up approach to describe a system. It is thus up to the modeller to find a relevant decomposition to study, and to decide the level of granularity of the model. If no interesting Multiplicity Principle is to be found in a certain decomposition in layers, maybe one could be found in another, as there is a multiplicity (pun intended) of decompositions. One may expect guidelines to facilitate the modeller's work. We leave this for future work.

If we restrict ourselves to the cases where the categories share the same objects, the superposition is very easy to compute and could be efficiently implemented with an adequate categorical framework.

In practical cases, such a model should be coupled with simulations or more quantitative tools. The language presented here is a starting point to conceptualise a resilient system and study its evolution. This framework is meant to allow for resilient design, but simulations will probably be needed to adjust the parameters.

Just as emergence could be formulated as a theorem in [9, Theorem 6, Section 7.2, Chapter 4, Part 1, page 139], we expect our layer model to allow for one or several theorem(s) on resilience, for example stating a few categorical conditions implying a form of resilience. Thus we need to endow the layer model with a categorical definition of resilience. We don't know yet whether such a definition should be system-agnostic.

We also hope this work will see applications in various fields of science, in the same vein as [28, 29, 30], and in particular applications to real systems in need of resilience.

Acknowledgements

This work was supported by MITACS Acceleration under Grant IT25652.

Disclosure statement

The authors report there are no competing interests to declare.

Biographical notes

Erwan Beurier has been a postdoctoral researcher at Polytechnique Montreal since 2021. He received a Master's degree in logic and mathematics in 2016 from Université Paris-Diderot, an Engineering Degree in 2017 and a Ph.D. in mathematics in 2020, both from IMT Atlantique, Brest, France. His main research consists of developing a mathematical theory of resilience, following previous work on category theory applied to biology. Other research interests include post-quantum cryptography and artificial intelligence.

Dominique Pastor was born in Cahors, France, in 1963. He graduated from Telecom Bretagne (Brest, France) in 1986 and from the University of Rennes (France) in 1997 (Ph.D.). From 1987 until 2000, he was with Thales. In particular, between 1990 and 1998, he was with Thales Avionics where his research concerned speech processing for applications to speech recognition systems embedded in military fast jet cockpits and, from 1998 to 2000, he was with Thales Nederland where he worked on the detection of radar targets in sea clutter. In September 2000, he joined Altran Technologies Nederland as a senior consultant. Since September 2002, he is with Institut Telecom, where he is currently Professor at Telecom Bretagne. His main research are robust statistical signal processing and mathematical approches to resilience.

Nora Boulahia-Cuppens is a full professor at Polytechnique Montréal in the Department of Computer and Software Engineering. She is the Deputy Director of Research at the Multidisciplinary Institute for Cybersecurity and Cyber Resilience (IMC2), and since 2021, she has been responsible for the CRITICAL chair on cybersecurity in the maritime sector. She obtained a Ph.D. from the National School of Aeronautics and Space and a 'Habilitation' to Direct Research from the University of Rennes 1 in France. Her research focuses on the three dimensions of cybersecurity: protection (Identity and Access Management, security policies, and protocols), defense (intrusion detection and responses), and resilience (moving target, functional diversification) in sectors of interest such as critical infrastructures, transportation, and 5G.

Frédéric Cuppens is a full professor at Polytechnique Montréal. Since 2023, he has been the director of the IMC2 and is responsible for the GEDAI institutional Chair on the identification, analysis, and automation of insider deviations and anomalies management. In 2002, he co-created a professional master's program in cybersecurity. From 2003 to 2020, he was a professor at IMT Atlantique and head of the IRIS team at Lab-STICC. He held the Cyber CNI Chair on Critical Infrastructure Cybersecurity. From 2014 to 2018, he led the Training Club at the Cyber Excellence Hub.

Data availability statement

No dataset was used, analysed or generated during this research.

References

[1] C. S. Holling, "Resilience and stability of ecological systems," *Annual Review of Ecology and Systematics*, vol. 4, no. 1, pp. 1–23, 1973. [Online]. Available: https://doi.org/10.1146/annurev.es.04.110173.000245

[2] L. McCubbin, “Challenges to the definition of resilience,” *109th Annual Meeting of the American Psychological Association*, August 2001. [Online]. Available: https://eric.ed.gov/?id=ED458498

[3] N. Doorn, P. Gardoni, and C. Murphy, “A multidisciplinary definition and evaluation of resilience: the role of social justice in defining resilience,” *Sustainable and Resilient Infrastructure*, vol. 4, no. 3, pp. 112–123, 2019. [Online]. Available: https://doi.org/10.1080/23789689.2018.1428162

[4] M. Barbeau, F. Cuppens, N. Cuppens, R. Dagnas, and J. Garcia-Alfaro, “Resilience estimation of cyber-physical systems via quantitative metrics,” *IEEE Access*, vol. 9, pp. 46 462–46 475, 2021.

[5] I. Linkov and A. Kott, *Fundamental Concepts of Cyber Resilience: Introduction and Overview*. Cham: Springer International Publishing, 2019, pp. 1–25. [Online]. Available: https://doi.org/10.1007/978-3-319-77492-3_1

[6] G. M. Edelman and J. A. Gally, “Degeneracy and complexity in biological systems,” *Proceedings of the National Academy of Sciences*, vol. 98, no. 24, pp. 13 763–13 768, 2001. [Online]. Available: https://www.pnas.org/content/98/24/13763

[7] R. Rosen, “A relational theory of biological systems II,” *The bulletin of mathematical biophysics*, vol. 21, pp. 109–128, June 1959.

[8] ——, “The representation of biological systems from the standpoint of the theory of categories,” *The bulletin of mathematical biophysics*, vol. 20, pp. 317–341, December 1958.

[9] A. Ehresmann and J.-P. Vanbremeersch, *Memory Evolutive Systems; Hierarchy, Emergence, Cognition*, 1st ed., ser. Studies in multidisciplinarity. Elsevier, 2007, vol. 4.

[10] ——, “Hierarchical evolutive systems: A mathematical model for complex systems,” *Bulletin of Mathematical Biology*, vol. 49, no. 1, pp. 13 – 50, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0092824087800332

[11] M. Barr and C. Wells, *Category Theory for Computing Science*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[12] S. MacLane, *Categories for the Working Mathematician*, 2nd ed., ser. Graduate Texts in Mathematics. Springer-Verlag, New York, 1998, vol. 5.

[13] D. I. Spivak, *Category Theory for the Sciences*. MIT Press, 2014.

[14] S. Awodey, *Category Theory*, 2nd ed., ser. Oxford Logic Guides. Oxford University Press, Oxford, 2010, vol. 52.

[15] N. A. Baas, “Hyperstructures as abstract matter,” *Advances in Complex Systems*, vol. 09, no. 03, pp. 157–182, 2006. [Online]. Available: https://doi.org/10.1142/S0219525906000732

[16] ——, “On structure and organization: an organizing principle,” *International Journal of General Systems*, vol. 42, no. 2, pp. 170–196, 2013. [Online]. Available: https://doi.org/10.1080/03081079.2012.728406

[17] ——, “On higher structures,” *International Journal of General Systems*, vol. 45, no. 6, pp. 747–762, 2016. [Online]. Available: https://doi.org/10.1080/03081079.2015.1118095

[18] ——, “On the philosophy of higher structures,” *International Journal of General Systems*, vol. 48, no. 5, pp. 463–475, 2019. [Online]. Available: https://doi.org/10.1080/03081079.2019.1584894

[19] ——, “On the mathematics of higher structures,” *International Journal of General Systems*, vol. 48, no. 6, pp. 603–624, 2019. [Online]. Available: https://doi.org/10.1080/03081079.2019.1615906

[20] N. A. Baas, A. C. Ehresmann, and J.-P. Vanbremeersch, “Hyperstructures and memory evolutive systems,” *International Journal of General Systems*, vol. 33, no. 5, pp. 553–568, 2004. [Online]. Available: https://doi.org/10.1080/0308107042000193534

[21] E. Beurier, D. Pastor, and R. Guitart, “Presentations of clusters and strict free-cocompletions,” *Theory and Applications of Categories*, vol. 36, no. 17, pp. 492–513, 2021. [Online]. Available: http://www.tac.mta.ca/tac/volumes/36/17/36-17abs.html

[22] A. C. Ehresmann and J.-P. Vanbremeersch, “Multiplicity principle and emergence in memory evolutive systems,” *Syst. Anal. Model. Simul.*, vol. 26, no. 1-4, pp. 81–117, Dec. 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=246318.246332

[23] E. Beurier, “Characterisation of organisations for resilient detection of threats,” Ph.D. dissertation, IMT Atlantique, Ecole Doctorale Mathstic, Brest, France, november 2020.

[24] D. Pastor, E. Beurier, A. Ehresmann, and R. Waldeck, “Interfacing biology, category theory and mathematical statistics,” in Proceedings *Applied Category Theory 2019,* University of Oxford, UK, 15-19 July 2019, ser. Electronic Proceedings in Theoretical Computer Science, J. Baez and B. Coecke, Eds., vol. 323. Open Publishing Association, 2020, pp. 136–148.

[25] R. Francis and B. Bekera, “A metric and frameworks for resilience analysis of engineered and infrastructure systems,” *Reliability Engineering and System Safety*, vol. 121, pp. 90–103, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832013002147

[26] A. Ehresmann and J. Vanbremeersch, “Memory evolutive systems: an application to an aging theory,” *Cybernetics and Systems*, pp. 190–192, 1993.

[27] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, “Monitoring and mapping with robot swarms for agricultural applications,” in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[28] J. de Lima do Rego Monteiro, J. E. Kogler, J. H. R. Ribeiro, and M. L. Netto, “On building a memory evolutive system for application to learning and cognition modeling,” in *Brain Inspired Cognitive Systems 2008*, A. Hussain, I. Aleksander, L. S. Smith, A. K. Barros, R. Chrisley, and V. Cutsuridis, Eds. New York, NY: Springer New York, 2010, pp. 19–39.

[29] M. Andreatta, A. Ehresmann, R. Guitart, and G. Mazzola, “Towards a categorical theory of creativity for music, discourse, and cognition,” in *Mathematics and Computation in Music*, J. Yust, J. Wild, and J. A. Burgoyne, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 19–37. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-39357-0_2

[30] J. Burgos-Salcedo, “A memory evolutive system-model of immunocompetence with applications to sars-cov-2 immunity,” *Sch Acad J Biosci*, vol. 4, pp. 77–84, 2022. [Online]. Available: https://saspublishers.com/media/articles/SAJB_104_77-84.pdf

[31] J. R. B. Cockett and S. Lack, "Restriction categories i: categories of partial maps," *Theoretical computer science*, vol. 270, no. 1-2, pp. 223–259, 2002.

[32] ——, "Restriction categories ii: Partial map classification," *Theoretical Computer Science*, vol. 294, no. 1-2, pp. 61–102, 2003.

[33] ——, "Restriction categories iii: colimits, partial limits and extensivity," *Mathematical Structures in Computer Science*, vol. 17, no. 4, pp. 775–817, 2007.

[34] K. Kunen, *Set theory - An introduction to independence proofs*, 7th ed., ser. Studies in logic and the foundations of mathematics. North-Holland Publishing Company, 1999, vol. 102.

A Some mathematical background

This appendix is divided into three subsections. We first briefly introduce the categorical material required for this paper (Section A.1) for the reader not familiar with category theory. Sections A.2 and A.3 make a detour, respectively, through the definition of the partial variants of functors and natural transformations (defined in Section A.1) and some foundational matters.

A.1 Crash course on category theory

In case the reader is not already familiar with category theory, references in the topic include [11], [12], [13] or [14] (we strongly recommend the last reference for mathematicians and the one before the last to other scientists). We recall here the basic notions used in this paper.

Definition A.1 (Category)**.** A *category* $\mathscr{C}$ is a tuple $(\mathrm{Ob}(\mathscr{C}), \mathrm{Mor}(\mathscr{C}), \circ, \mathrm{Id}(\mathscr{C}))$ such that:

1. $\mathrm{Ob}(\mathscr{C})$ is a collection whose elements are called *objects*, denoted by single letters C, C', C_0, C_1...
2. $\mathrm{Mor}(\mathscr{C})$ is a collection whose elements are called *morphisms*, or, interchangeably, *arrows*. A morphism (or *arrow*) c is denoted by an arrow linking two objects $c: C \to C'$; C is here called the *domain* of c, while C' is its *codomain*. We will often write $\mathrm{dom}(c) = C$ and $\mathrm{cod}(c) = C'$.
3. $\circ : \mathrm{Mor}(\mathscr{C}) \times \mathrm{Mor}(\mathscr{C}) \to \mathrm{Mor}(\mathscr{C})$ is an associative composition law. If $c : C \to C'$ and $c' : C' \to C''$ are two arrows such that $\mathrm{dom}(c') = \mathrm{cod}(c)$, then their composition is $c' \circ c : C \to C''$.
4. $\mathrm{Id}(\mathscr{C})$ is a subcollection of $\mathrm{Mor}(\mathscr{C})$ consisting only of those arrows $\mathrm{id}_C : C \to C$ such that, for all $c : C \to C'$, $c \circ \mathrm{id}_C = c$ and for all $c' : C' \to C$, $\mathrm{id}_C \circ c' = c'$. These arrows are called *identities*, they are basically the units for the composition law, and there is exactly one identity per element in $\mathrm{Ob}(\mathscr{C})$.

A category is basically a graph with identities, and whose any two consecutive arrows compose. In all generality, $\mathrm{Ob}(\mathscr{C})$ and $\mathrm{Mor}(\mathscr{C})$ can be bigger than sets. When they are both sets, as will be the case in this paper, we call the category $\mathscr{C}$ *small*. Small categories are generally preferred as they are more convenient to use in mathematics.

Usual categories include ***Sets***, the category of sets and functions, and most "structured sets" such as ***Groups***, the category of groups and group morphisms, or ***Rings***, the category of rings and ring morphisms.

Categories come with their own notion of morphisms:

Definition A.2 (Functor)**.** Given two categories $\mathscr{C}$, $\mathscr{X}$, a *functor* $F : \mathscr{C} \to \mathscr{X}$ is a mapping $F : \mathrm{Ob}(\mathscr{C}) \cup \mathrm{Mor}(\mathscr{C}) \to \mathrm{Ob}(\mathscr{X}) \cup \mathrm{Mor}(\mathscr{X})$ such that:

1. For all $C \in \mathrm{Ob}(\mathscr{C})$, $F(C) \in \mathrm{Ob}(\mathscr{X})$.
2. F preserves domains and codomains: for all $c : C \to C' \in \mathscr{C}$, $F(c) : F(C) \to F(C')$. In particular, $F(c) \in \mathrm{Mor}(\mathscr{X})$.
3. F preserves identities: for all $\mathrm{id}_C : C \to C$, $F(\mathrm{id}_C) = \mathrm{id}_{F(C)}$.
4. F preserves compositions: for all $c : C \to C'$ and $c' : C' \to C'' \in \mathscr{C}$, $F(c' \circ c) = F(c') \circ F(c)$.

Functors are sometimes called *diagrams*. Those two words have the same mathematical definition, but they are used in different contexts: when using the word *diagram* for a functor $F : \mathscr{C} \to \mathscr{X}$, we generally mean that the main topic of interest is $\mathscr{X}$ and its "internal" properties. It also suggests that $\mathscr{C}$ is "smaller" in size than $\mathscr{X}$ and that, by studying F, we consider a small part of $\mathscr{X}$. This is why both words generally coexist in the same works.

Functors are morphisms between categories, it is thus natural to define $\boldsymbol{Cat}$, the category of (small) categories and functors between them. Note that $\boldsymbol{Cat}$ is not small itself, so $\boldsymbol{Cat}$ does not contain itself, phew.

In this paper, we use the convention that functors are written with the letters F, F' and diagrams are written with the letters $D : \mathscr{D} \to \mathscr{C}$, $E : \mathscr{E} \to \mathscr{C}$. Diagrams to $\boldsymbol{Cat}$ will rather be noted as $P : \mathscr{P} \to \boldsymbol{Cat}$.

One can use two diagrams to create a new category. Here is the recipe:

Definition A.3 (Comma-category)**.** Consider two diagrams $D : \mathscr{D} \to \mathscr{C}$ and $E : \mathscr{E} \to \mathscr{C}$ to the same category $\mathscr{C}$. The *comma-category* $(D \downarrow E)$ is the following category:

Objects: The objects are the triples (d, g, e) such that $d \in \mathscr{D}$, $e \in \mathscr{E}$ and $g : D(d) \to E(e) \in \mathscr{C}$

Morphisms: The arrows are the pairs $(a, b) : (d, g, e) \to (d', g', e')$ such that $a : d \to d' \in \mathscr{D}$, $b : e \to e' \in \mathscr{E}$ and the following diagram commutes:

$$\begin{tikzcd}
D(d) \arrow[r, "D(a)"] \arrow[d, "g"'] \arrow[dr, phantom, "\checkmark"] & D(d') \arrow[d, "g'"] \\
E(e) \arrow[r, "E(b)"'] & E(e')
\end{tikzcd}$$

We even have morphisms between functors:

Definition A.4 (Natural transformation)**.** Given two functors $F, F' : \mathscr{C} \to \mathscr{X}$, a *natural transformation* $f : F \to F'$ is a collection

$$\alpha = \big(f_C : F(C) \to F'(C)\big)_{C \in \mathscr{C}}$$

of arrows of $\mathscr{X}$ such that, for all $c : C \to C' \in \mathscr{C}$, the following diagram commutes:

$$\begin{tikzcd}
F(C) \arrow[r, "F(c)"] \arrow[d, "f_C"'] \arrow[dr, phantom, "\checkmark"] & F(C') \arrow[d, "f_{C'}"] \\
F'(C) \arrow[r, "F'(c)"'] & F'(C')
\end{tikzcd}$$

or explicitly, such that, for all $c : C \to C' \in \mathscr{C}$, $F'(c) \circ f_C = f_{C'} \circ F(c)$.

Definition A.5 (Functor category)**.** Given two categories $\mathscr{C}$ and $\mathscr{X}$, we define the *functor category* $\boldsymbol{Func}(\mathscr{C}, \mathscr{X})$, also written $\mathscr{X}^{\mathscr{C}}$, as the category of functors $\mathscr{C} \to \mathscr{X}$ and natural transformations between them.

Definition A.6 (Diagonal functor)**.** For every pair of categories $\mathscr{C}$ and $\mathscr{X}$, the *diagonal functor* is defined by:

$$\Delta^{\mathscr{C}}_{\mathscr{X}} : \left\{ \begin{array}{ccc} \mathscr{X} & \longrightarrow & \boldsymbol{Func}(\mathscr{C},\mathscr{X}) \\ X & \longmapsto & \Delta^{\mathscr{C}}_{\mathscr{X}}(X) : \left\{ \begin{array}{ccc} \mathscr{C} & \longrightarrow & \mathscr{X} \\ C & \longmapsto & X \\ c & \longmapsto & \mathrm{id}_X \end{array} \right. \\ x & \longmapsto & (x : X \to X')_{C \in \mathscr{C}} \end{array} \right.$$

The diagonal functor $\Delta^{\mathscr{C}}_{\mathscr{X}}$ basically creates constant functors $\Delta^{\mathscr{C}}_{\mathscr{X}}(X)$. It is useful in the following definition:

Definition A.7 (Cocone)**.** Let $D : \mathscr{D} \to \mathscr{C}$ be a diagram to a category $\mathscr{C}$. A *cocone* $\alpha : D \to C$ is a natural transformation $\alpha : D \to \Delta^{\mathscr{D}}_{\mathscr{C}}(C)$. More explicitly, it is a collection of arrows $(D(d) \to C)_{d \in \mathscr{D}}$ such that, for all $a : d \to d'$, the following diagram commutes:

$$\begin{tikzcd} D(p) \arrow[r, "D(a)"] \arrow[dr, "\alpha_C"'] & D(d') \arrow[d, "\alpha_{C'}"] \\ & C \end{tikzcd}$$

The *peak of* $\alpha : D \to C$ is the object C.

Certain cocones are of special interest:

Definition A.8 (Colimit)**.** Let $D : \mathscr{D} \to \mathscr{C}$ be a diagram to a category $\mathscr{C}$. The *colimit of* D, if it exists, is a cocone $\mathrm{Colim}(D) : D \to cD$ such that:

1. cD is an object of $\mathscr{C}$
2. For all other cocone $\alpha : D \to C$, there exists a unique $u : cD \to C$ such that, for all $d \in \mathscr{D}$, the following diagram commutes:

$$\begin{tikzcd} D(d) \arrow[d, "\mathrm{Colim}(D)_d"'] \arrow[dr, "\alpha_d"] & \\ cD \arrow[r, "u"'] & C \end{tikzcd}$$

   This unique arrow u, or the property that it exists, is referred to as the Universal Mapping Property, or UMP.

Colimits are an example of universal constructions, in the sense that all other similar constructions basically boil down to them; here, cocones boil down to colimits. Universal constructions are common in category theory, however in this article we will only encounter colimits.

For more details, and more constructions, the interested reader may refer to the above-mentioned introductory works [11, 12, 13, 14].

A.2 Partiality

In this paper, we consider partial functors.

Definition A.9 (Partial functor)**.** Let $\mathscr{C}$ and $\mathscr{C}'$ be two small categories.

A *partial functor* $F : \mathscr{C} \to \mathscr{C}'$ is a functor $F : \mathscr{C}_0 \to \mathscr{C}'$ such that $\mathscr{C}_0$ is a subcategory of $\mathscr{C}$.

Such partial functors define a restriction structure on the category of small categories in the sense of [31, 32, 33]. Denote by ***PartCat*** the category of small categories and partial functors.

We also define partial natural transformations:

Definition A.10 (Partial natural transformations)**.** A *partial natural transformation* is a natural transformation in the usual sense, between two diagrams $F, G : \mathscr{C} \to \boldsymbol{PartCat}$.

A partial natural transformation is *partial* due to its components being (potentially) partial functors. It is not the equivalent of a natural transformation between partial functors (which would be another notion that we do not need in this paper).

A.3 On the finitude of categories

In this paper, foundational issues may arise at some point with the use of $\boldsymbol{Cat}$. We always assumed that $\boldsymbol{Cat}$ behaved like a small category, which is probably not true. This section explains a workaround that we can use to fix this misuse of language.

The goal of this paper is to define a categorical language for reasoning about real-world systems. Real-world systems are finite in size, and in components. One may conclude that a system should be represented by a finite category, which is not the case, because of the composition of arrows. If there is one loop of non-idempotent arrows, then the category will have a countable infinite number of arrows.

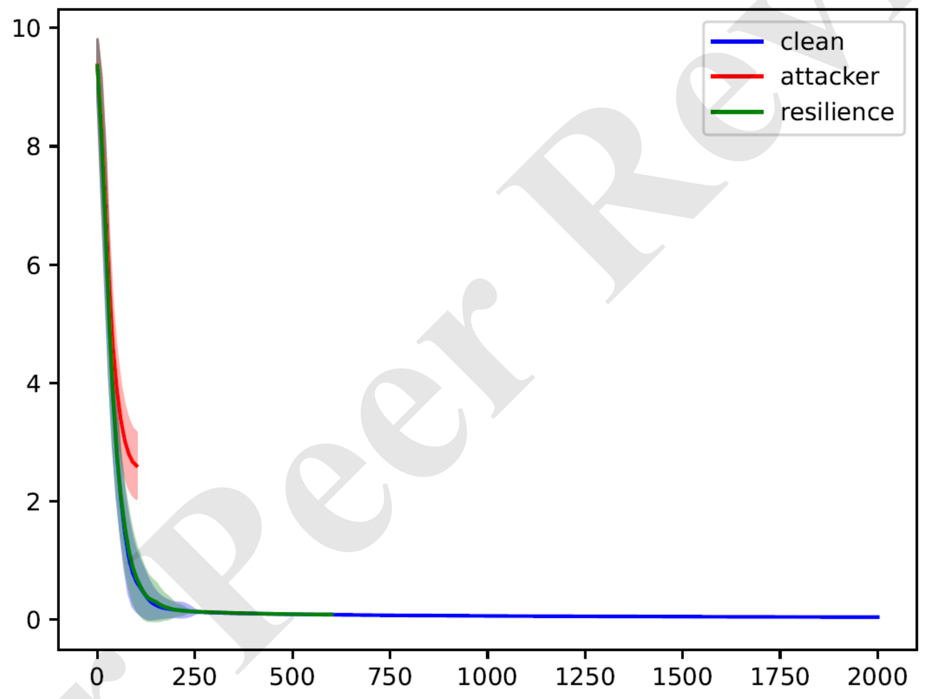
Thus, in this paper, we will not consider a hypothetical category $\boldsymbol{FinCat}$ of finite categories, but rather, a category of finitely-generated categories. However, such a category will not be small, and again, for the purposes of this article, we want to consider only a small category of finitely-generated categories. To do that, we use set-theory to consider only the finitely-generated categories that belong to the set V_{ω_1} of the cumulative hierarchy of the Von Neumann universe. Maybe smaller V_α's could suffice, but to avoid perfectly tailoring all the mathematical objects, let's consider a bigger one that will necessarily do the job. The category of finitely generated categories contained in V_{ω_1} is denoted as $\boldsymbol{Cat}_{\omega_1}$.

For more information on the Von Neumann hierarchy, refer to [34, Chapter III, §2, pages 95-98], where the hierarchy is described in more details under the term of *well-founded sets*.

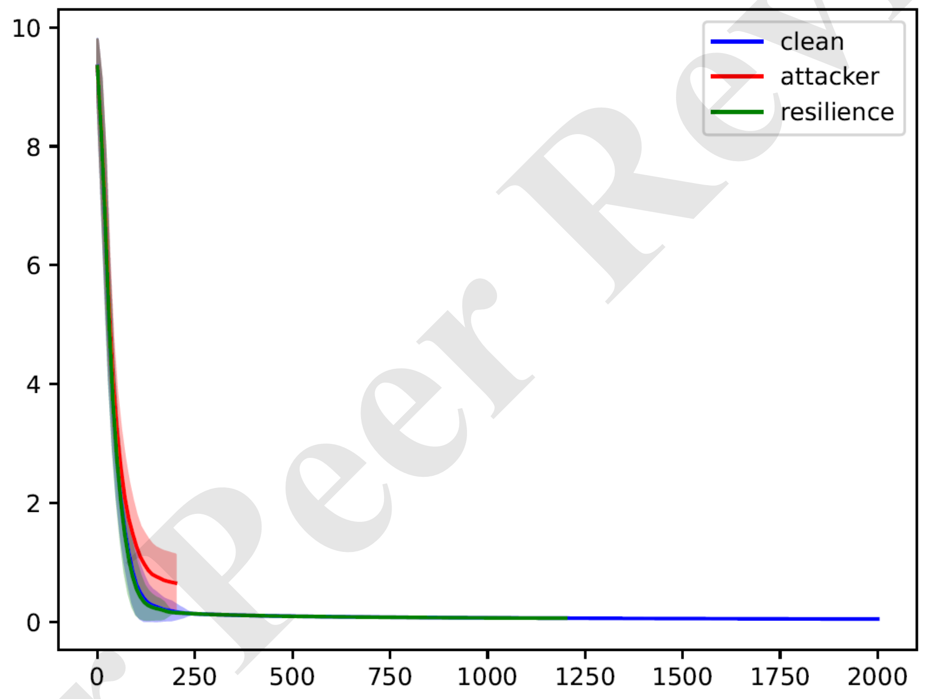
However, for the sake of readability, in order to limit the use of confusing new notation, and to avoid set-theoretical discussions, we used the notation $\boldsymbol{Cat}$, which is standard and well-known, instead of $\boldsymbol{Cat}_{\omega_1}$.

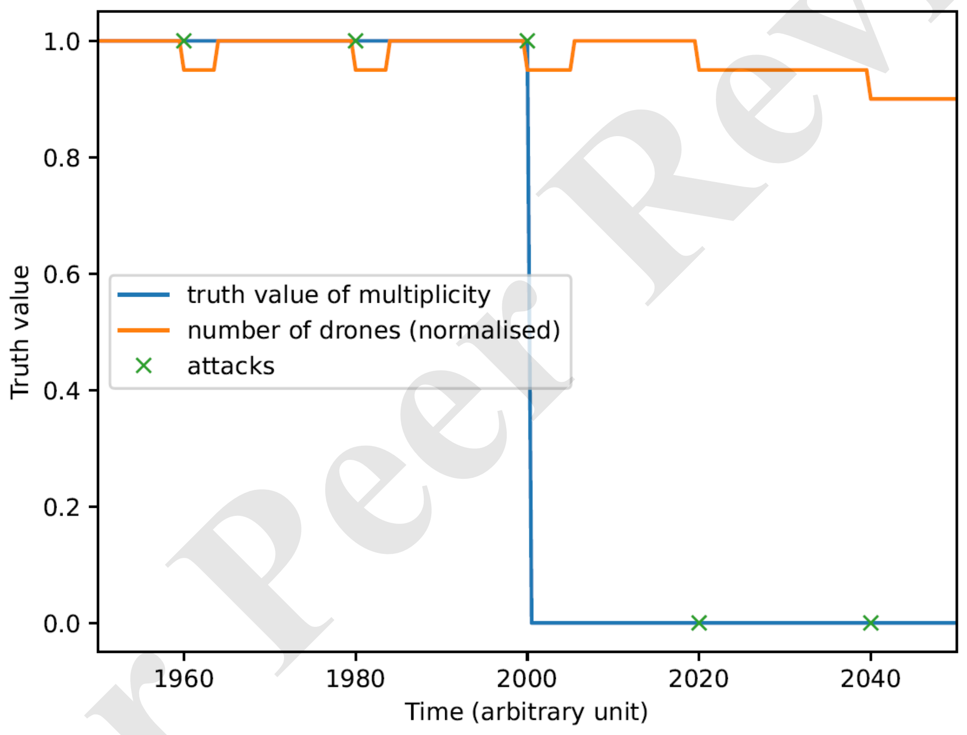
List of Tables

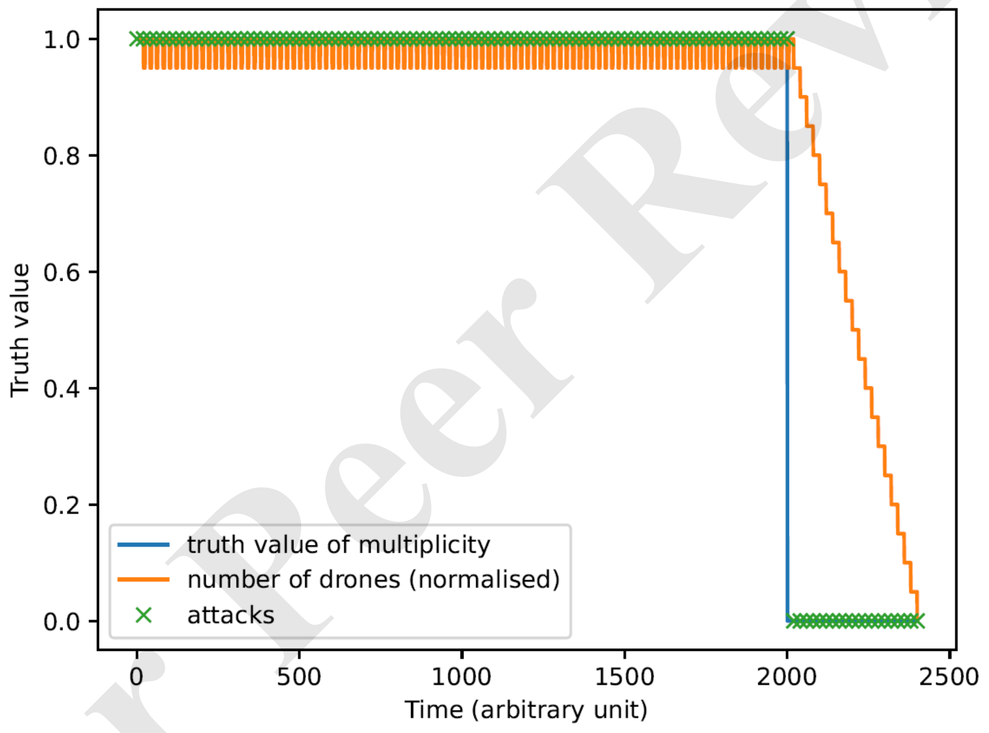
List of Figures

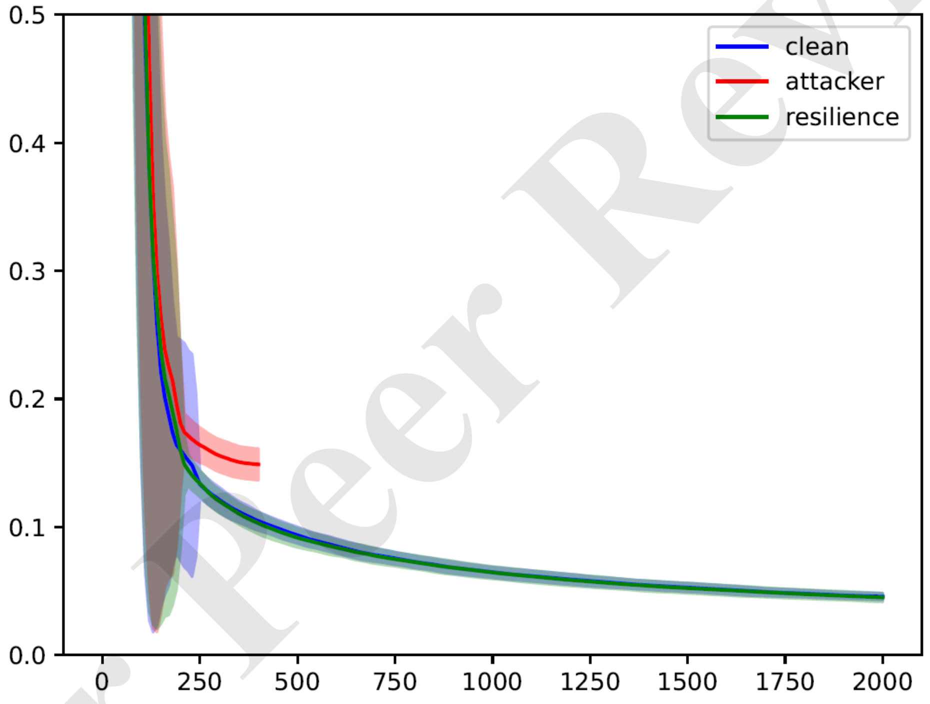


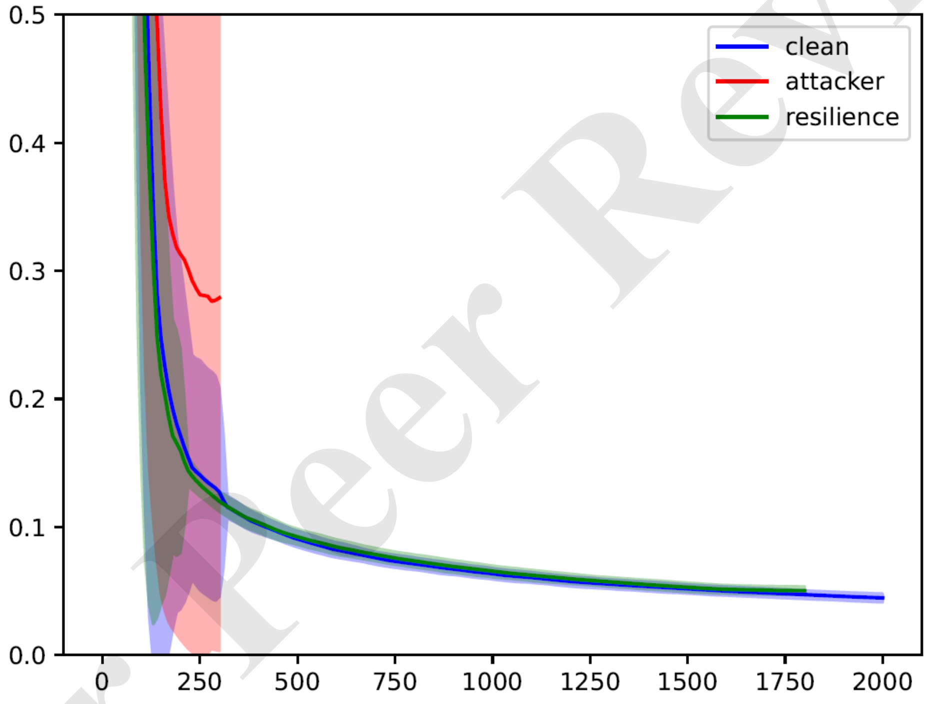
clean
attacker
resilience


10
8
6
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
clean
attacker
resilience


truth value of multiplicity
number of drones (normalised)
attacks
Truth value
Time (arbitrary unit)
1.0
0.8
0.6
0.4
0.2
0.0
1960
1980
2000
2020
2040


1.0
0.8
0.6
0.4
0.2
0.0
Truth value
0
500
1000
1500
2000
2500
Time (arbitrary unit)
truth value of multiplicity
number of drones (normalised)
attacks


0.5
0.4
0.3
0.2
0.1
0.0
0
250
500
750
1000
1250
1500
1750
2000
clean
attacker
resilience



0.5
0.4
0.3
0.2
0.1
0.0
0
250
500
750
1000
1250
1500
1750
2000
clean
attacker
resilience