The Etiology of Cybersecurity

Michele Ambrosi[1], Francesco Beltramini[1], Federico De Meo[1], Olivero Nardi[1], Mattia Pacchin[1], and Marco Rocchetto

V-Research
Verona, Italy
knowledgezero@v-research.it

Abstract. The objective of this research is to lay the foundations for the development of a scientific theory that determines (all and only) the possible insecure and secure configurations of any abstract system. We claim that cybersecurity weaknesses (i.e. errors) are at the beginning of the causality chain that leads to cybersecurity attacks. We, then, formulate a scientific (falsifiable by experiments) hypothesis that we use to predict all the weaknesses in the architectural design of an abstract system. The mathematical formulation of our hypothesis is based on a correlation between the epistemological concepts of facts, beliefs, and assertions with the engineering concepts of requirements, functional architectures, and channels. Ultimately, our hypothesis allows for the definition of a mathematical formula which describes the cybersecurity of a system. We implemented a prototype cybersecurity risk assessment tool that, based on our hypothesis, predicts the weaknesses in a UML model of a (cyber-physical) system.

Keywords: Risk Management, Cyber-Physical Systems, Risk Assessment, Security Framework

1 Introduction

tofix:

1. Herley's work (1 paragraph)
2. Implications on Risk Assessment (no theory implies no complete set of weakness which, in turn, implies no risk assessment)
3. 1 page: contributions

Context and motivation A *scientific theory* is an explanation of a phenomenon such that the explanation follows the scientific method. The *scientific method* is an *empirical* method that aims at mitigating potential fallacies in theories. Karl Popper famously argued (e.g. in [**Popper1959logic**]) that a scientific theory can never be verified but only falsified, that a theory should not be conceived by

using the principle of induction[1], and that empirical experiments should be considered as the only evidence to support the non-falseness of a scientific theory. **toremove:** If we consider a natural (real-world) phenomenon, such as gravity, we can think of a scientific theory as an *abstract* mathematical formula (or a set of formulas) that defines axiomatically a certain (natural) phenomenon in a specific algebraic structure. While such a theory can be proved to be correct within the mathematical setting it has been conceived in (e.g. with respect to other axioms of the algebraic structure where it has been postulated, or with respect to other background theories), no proof can be carried out to verify that the theory properly abstracts the phenomenon it describes (i.e. that the theory is a sound or complete approximation of the real phenomenon). Similarly, the results of the application of a scientific theory (i.e. its predictions) can be verified in the abstract mathematical structure, but how can we be sure that the theory correctly characterizes the phenomenon it describes? Popper's famous "demarcation principle" draws a line between scientific and pseudo-scientific theories by proposing that a scientific theory can be considered acceptable as long as no empirical experiment falsify it. For example, a theory of gravity, in its mathematical setting, can predict how objects attract each other, and those predictions can be tested by empirical experiment. If the theory correctly represents the phenomenon, no empirical experiment should falsify the predictions of that theory.

In [**Herley2016unfalsifiability**], Cormac Herley explores what he calls "an asymmetry in computer security", which he defines as follows: "Things can be declared insecure by observation, but not the reverse. There is no observation that allows us to declare an arbitrary system or technique secure". With security, Herley only focuses on cybersecurity (we also use security and insecurity, in this paper, only to refer to cyber-insecurity and cybersecurity) and his intuition is that there is no scientific theory that can predict the cybersecurity of a system, nor a theory that can predict all possible insecurities of a system (which, by negation, may be used as a theory of cybersecurity). Herley then uses this argument to show that "claims that any measure is necessary for security are empirically unfalsifiable". **toremove:** An example is whether or not password masking makes a system secure. Password masking prevents a complete leakage of the password, and a system, in which password masking is implemented, is believed to be secure because an attacker won't be seeing the password by "shoulder surfing". This reasoning is circular and it is always true that a system in which password masking is implemented is a system in which a password is masked. No experiment can ever falsify this statement[2]. Given that any theory that is not

[1] Albert Einstein wrote to Popper: "[...] and I think (like you, by the way) that theory cannot be fabricated out of the results of observation, but that it can only be invented." [**Popper1959logic**]

[2] In other words, one cannot design an experiment to show that password masking is not secure or unnecessary, and a correlated problem is that this may lead to a scenario where more measures are added but there is no good way of saying which measures are superflous.

falsifiable by an empirical experiment is well known[3] to be nonscientific (i.e. unfalsifiability is a fallacy of a theory), Herley concludes that there is no scientific theory supporting this type of cybersecurity measures; generalizing, cybersecurity seems to lay in the realm of pseudo-sciences [**Herley2016usenixvideo**]. Herley, e.g. in [**Herley2017justifying**], discusses the implications of a nonscientific approach to cybersecurity, and highlights the tremendous impact on the scientific research and engineering of systems. While the criticism is investigated in [**Herley2016unfalsifiability**], no solution is provided nor envisioned.

Contributions The goal of this paper is to lay the foundations of a scientific cybersecurity theory. We consider the problem raised by Herley not confined to "computer security" but rather we reason on any abstract system (so that our scientific hypothesis[4] may be tested in any sound implementation such as networks, mechanical, cyber, or cyber-physical system, or even a single computer or a single device such as a hard-drive).

Instead of starting from reasoning on what makes a system secure or insecure, we reason on what causes insecurities. We focus on insecurities only caused by the exploitation of cybersecurity attacks, and we assume that achieving cybersecurity means preventing all those attacks from being exploitable or exploited. Our hypothesis is that *cybersecurity attacks are only caused by the presence of errors in the design or implementation of a system* (i.e. cybersecurity weaknesses). **toremove:** For example, a design or implementation weakness in a sanitization function may allow an attacker to perform SQL-injections. If weaknesses are necessary to enable cybersecurity attacks, a theory that predicts all potential weaknesses can be effectively used to predict the (in)security of a system. Therefore, a possible answer to the initial question can be that (definition) cybersecurity is the absence of attacks and (claim) the absence of weaknesses implies cybersecurity. This, however, raises another question: "which are all the errors/weaknesses in (the engineering of) a system that make it possible for a cybersecurity attack to happen?". More rigorously, which scientific theory predicts all possible weaknesses that causes cybersecurity attacks?

With our approach, a list of weaknesses emerges from the mathematical formulation of a system in a framework called $\mathcal{ABF}$-framework, and predicts 4 main classes of weaknesses. Those classes of weaknesses are used to calculate all the insecurity configurations of all the components of a system, obtaining a precise estimation of all potential cybersecurity-related risks in any given system. Our hypothesis can be falsified by means of experiments, testing if all the predicted weaknesses are present in the system under test, or testing if other (not predicted

[3] "A theory which is not refutable by any conceivable event is nonscientific. Irrefutability is not a virtue of a theory (as people often think) but a vice." – Karl Popper, Conjectures and Refutations [**popper1962conjectures**].

[4] In the reminder of this paper, we will use the word hypothesis to refer to "scientific hypothesis" as a proposed scientific theory that has not gone through an extensive series of tests. Instead, we use "logical theory" to refer to a set of formal logical axioms.

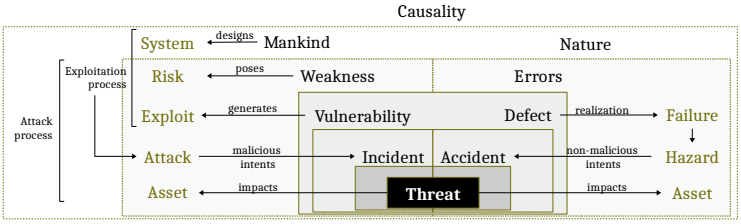

Fig. 1. Overview of Keywords Related to Cybersecurity and Safety

by our hypothesis) weaknesses are present. In fact, if any cybersecurity weaknesses were to be found in a system and not predicted by our hypothesis, the hypothesis could be declared incomplete. If a cybersecurity weakness would be predicted by our hypothesis but found to be impossible to realize, our hypothesis could be declared as wrong.

toremove:

Structure We start, in Section 2, by detailing the problem statement, reporting a literature review on the main concepts and definitions related to cybersecurity. We formulate a cybersecurity hypothesis in Section 3; which we use to propose a mathematical formulation of our hypothesis on system cybersecurity in Section 4. In Section 4.3, we apply our hypothesis for the tool-assisted cybersecurity risk assessment of a Cyber-Physical System (CPS), with an ad-hoc example. This application shows how our hypothesis can be used to predict all of the possible cybersecurity weaknesses of a system, allowing the falsification of our hypothesis.

2 Literature Review

toremove:

In most of the natural languages the concepts of safety and security are not syntactically differentiated and both terms (safety and security) are expressed by the same word, e.g. "sicurezza" in Italian. A semantic distinction between safety and (cyber)security is correlated to a belief[5] that safety deals with *accidents* (i.e. an unfortunate incident) posed by the natural environment (e.g. natural events such as wearing of hardware components), whereas cybersecurity deals with *incidents* posed by mankind (e.g. attackers and bugs). The fundamental difference between nature and mankind (and, in turn, between safety and cybersecurity) is believed to be on the different intents (incidents are intentional, whereas accidents are not) of the causes that generates a threat. Namely, nature is believed not to have malicious intents (but unfortunate causes-effects), whereas threats generated by mankind are malicious[6]. This conclusion seems to be against a

[5] A belief has to be intended as a proposition which is supposed to be true but no conclusive evidence makes it known to be true.

[6] Of course, logical flaws or bugs may be introduced by other means (e.g. ignorance) without explicit malicious intents, but the exploitation of those flaws is considered

general formulation of a cybersecurity theory; how can we define a theory that predicts what a human being will do?

Cybersecurity attacks seem to be related to the creativity of the attacker and thus unpredictable.

Currently, the most complete understanding of insecurity issues is stored into a network of databases of weaknesses (e.g. CWE [**MITRE2020CWEresearch**]), vulnerabilities (e.g. CVE [**CVE**], NVD [**NIST2020NVD**]), and attacks (e.g. CAPEC [**MITRE2020CAPEC**], ATT&CK [**MITRE2020ATTACK**]). Those insecurity issues can be related to the violation of one or more requirements (explicit or implicit) in the specification, design or implementation of a system. The correlation between insecurity flaws and cybersecurity requirements has been used to define standards such as the IEC 62443-1-3 (the Industrial communication networks - Network and system security – Part 3-3: System security requirements and security levels) which defines requirements as "confidentiality of information in transit/at-rest". More generally, the idea of defining cybersecurity requirements as properties of a system was initially defined in 1970s with the CIA triad (Confidentiality, Integrity, Availability) and refined over the decades introducing related concepts such as authenticity or non-repudiation, or introducing new ones such as "responsibility" in the RITE approach (see [**Samonas2014cia**] for an overview of the evolution of the CIA triad). The link between cybersecurity requirements and vulnerabilities is reported in the NVD databases by the CVSS [**Mell2007CVSS**] scoring system. The CVSS evaluates of the severity of a vulnerability by means of different metrics (such as attack complexity and user interaction) and quantitatively evaluates the impact on the CIA triad. While cybersecurity requirements, weaknesses, vulnerabilities, and attacks have been extensively studied and implemented both in academia and industry to provide tools for the testing or verification of systems, no scientific falsifiable theory correlates cybersecurity requirements to necessary and sufficient conditions (e.g. mitigations) to declare a system secure [**Herley2016unfalsifiability**]. Nonetheless, the extensive body of literature has scientific foundations, for example, providing formal frameworks for the verification of properties for cybersecurity. As a driver for our argumentation, we start by reviewing the key concepts in the cybersecurity domain.

2.1 Terminology

toremove: Most of the safety-preserving principles in the field of engineering of safety-critical cyber-physical systems (such as elevators and aircraft), upon which safety requirements are defined (e.g. in standards such as the IEC 61508 or 61511 [**IEC201761511**]), are based on empirical tests and measurements. While reasoning by induction based on the empirical observation should be avoided in general, since it may easily lead to false beliefs [**Popper1959logic**], inducing general principle by means of tests is often justified by the supposed impossibility

(for now, and detailed afterwards in the article) malicious, and thus we consider any vulnerability to be malicious even if it is due to the lack of skills.

of defining a theory that correctly predicts failures. A failure of a wire due to environment (e.g. due to humidity, dust, heat &c) is defined from empirical evidences and processes have been standardized to test qualities of hardware components. This process completely breaks down when a malicious environment (i.e. an attacker) is considered instead of the (supposedly honest and predictable) natural environment. Therefore, the same approach that is in use for safety, seems not to be applicable for cybersecurity (e.g. for cybersecurity testing). An overview on the aforementioned aspects of safety and cybersecurity is depicted in Figure 1 and is used as **tofix:** We provide a baseline for a definition of the terms that structure our current understanding of cybersecurity.

tofix: item to be moved in the itemize list below

- *Vulnerability*, as defined in [**cnssi20104009**] (and adopted in [**nist2013800-53**]), is a "weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source". **toremove:** On the one hand, the definition is broad to enclose as much causes (that generates a vulnerability) as possible, on the other hand the term vulnerability should have a complete and sound definition, so that no other causes (e.g. other sources) but the ones in the definition are responsible for a vulnerability. Furthermore, the term "threat sources" used in the definition in [**cnssi20104009**] may be identified with both Nature and Mankind, not differentiating between safety and security.

toremove: As depicted in Figure 1, a vulnerability does not necessarily become a threat for the system, unless exploited "through a channel that allows the violation of the security policy [...]" [**cnssi20104009**]. For example, a software or procedure that takes advantage of the vulnerability causing an *attack* to the system may result in several correlated incidents and threats. The process of exploitation of a defect as a vulnerability is reported in Figure 1 such that the difference between exploit and failure, and attack and accident, is to be found just in the maliciousness of the intents that causes this process (i.e. excluding the intent, the terms are just syntactic transformation from a vulnerability to defect, from accident to incident).

- *Weakness.* The definition given by the MITRE in [**MITRE2020CWEweakness**] of weakness is: " a type of mistake that, in proper conditions, could contribute to the introduction of vulnerabilities within that product. This term applies to mistakes regardless of whether they occur in implementation, design, or other phases of a product lifecycle." A vulnerability, such as those enumerated on the Common vulnerabilities and Exposures (CVE) List, is a mistake that can be directly used by an attacker to gain access to a system or network. The definition is circular if we interpret the word "error" and "mistake" with the same semantics: a weakness is an error that leads to a vulnerability and a vulnerability is a mistake which, in turn, is a weakness. The only difference between a weakness and vulnerability seems to be that one can consider weakness as a ground term and state that a vulnerability is caused by a weakness.

- **toremove:** *Causality* refers to the causality principle; defined in [**Spirkin1983Dialectical**] as "Causality is a genetic connection of phenomena through which one thing (the cause) under certain conditions gives rise to, causes something else (the effect). The essence of causality is the generation and determination of one phenomenon by another. In this respect, causality differs from various other kinds of connection, for example, the simple temporal sequence of phenomena, of the regularities of accompanying processes".
- An *Exploit* "[...] (from the English verb to exploit, meaning to use something to one's own advantage) is a piece of software, a chunk of data, or a sequence of commands that takes advantage of a bug or vulnerability to cause unintended or unanticipated behavior to occur on computer software, hardware, or something electronic (usually computerized)." [**wiki-exploit**].
- An *Attack*, as defined by the International Standard ISO/IEC 27000, is an "attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset"; where an *Asset* is "anything that has value to the organization". **toremove:** Furthermore, we note that for the purpose of this article, we do not want to focus on a specific organization or business to define asset but, in general, on any abstract organization (e.g. a company or a society). We do not consider ethical hackers as attacking a system. In fact, we consider the term *hack* as non-malicious (as, e.g. in [**Stallman2002hacker**]).
- A *Threat*, as defined in [**cnssi20104009**], is "Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service".
- *Defect*: "anything that renders the product not reasonably safe" [**Robinson2019writing**] (i.e. a characteristic of an object which hinders its proper usability).
- *Failure*, as defined in [**Merriam2020failure**], is "a state of inability to perform a normal function". The term is structured and detailed in [**cnssi20104009**, **iet2017glossary**], but relying on an abstract notion of failure without a specific definition.
- *Hazard*, "a potential source of harm" [**iet2017glossary**].

toremove: Our literature review shows that most of the definitions relate insecurity to dis-honesty (also called maliciousness or adversarial) of an agent (often called adversary or attacker). In fact, weaknesses become vulnerability if an attacker exploits them in an attack. This, however, just shifts the problem of defining what cybersecurity is as the problem of defining what a dishonest agent can or cannot do in a system, where an agent is any virtual or physical entity of the system or using the system (e.g. a device, a software, or a human being), and dishonesty is not necessarily related to malicious motivation but also to incompetence or lack of skills. As in the Dolev-Yao attacker model[7]

[7] The Dolev-Yao attacker model can be considered as abstract model of an attacker that can be used for the analysis of protocol specifications.

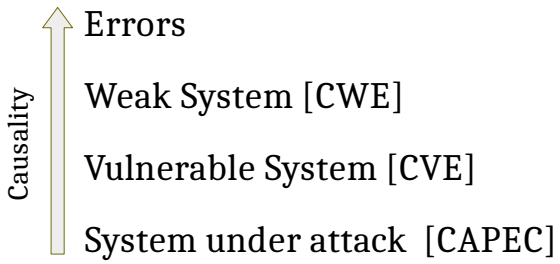

Fig. 2. Etiology of cybersecurity

[**Dolev1983security**], we may correlate "being dishonest" to "not following the intended behavior/rules". In the case of a generic system, a dishonest agent is, therefore, any agent that doesn't follow the intended behavior (or functionality or logic) of the system. For example, a software can be considered an agent of the system and whenever it has a bug, it can be exploited causing an incident. This may lead to the conclusion that the dishonest behaviors of agents cannot be defined in general (e.g. due to the heterogeneity of agents and systems) and that a huge repository of dishonest behaviors should be kept as definition, such as CAPEC [**MITRE2020CAPEC**]; or that dishonesty of an agent with respect to a cybersecurity protocol should be defined as a number of predefined actions as in [**Turuani2006clatse**, **Basin2005ofmc**, **Armando2016satmc**, **Rocchetto2017interpolation**] (to name a few)[8]. As depicted in Figure 2, in order to define a theory on cybersecurity we may say that the presence of vulnerabilities is a necessary condition in order to cause an attack in the system. Those vulnerabilities are, in turn, caused by the presence of weaknesses in the system. Weaknesses are errors in the design or implementation of a system. Therefore, a theory on cybersecurity should first predict the errors in a system design.

3 A Cybersecurity Hypothesis in the $\mathcal{ABF}$-framework

In order to address the problem raised by Herley, we shall define how to distinguish between a secure and an insecure system. **toremove:** While most of the literature have correlated the problem of in-security to the maliciousness of agents interacting with the system, we show that cybersecurity doesn't seem to stem from a malicious nature but, rather, insecurity raises from the lack of well-defined cybersecurity requirements for the development process of a system. We argue that the high number of cybersecurity vulnerability reported today are simply the realization of potential system configurations, which deviate from the *intended (or nominal) behavior* because the nominal behavior of the system is

[8] In formal protocol verifiers, in the DY model, the attacker has a set of hard-coded inference rules that defines how the attacker deduces new messages based on the observed message exchange.

not precisely defined in the specification at the very early stages of the engineering process. As a reference, we consider the following steps of the engineering process and our objective is to be able to predict the cybersecurity requirements instead of axiomatically assume them.

1. *System Specification*, where the *functional and physical requirements* are defined.
2. *Architecture Design*, where the specification is structured into *functional and physical architectures*.
3. *Cybersecurity Risk Assessment*, where the potential *weaknesses* (errors) are identified and, consequently, *cybersecurity requirements* are captured.

Given that, in our investigation, we changed the focus from the attacker to the potential designs of a system, we start by defining a general framework for the definition of a system. On top of this framework we will identify system weaknesses as potential design errors.

toremove:

3.1 Epistemology and Multi-Agent Systems

The ISO/IEC/IEEE 15288:2015 (System Life Cycle Processes) provides a definition of system as "A combination of interacting elements organized to achieve one or more stated purposes."[**ISO201515288**]. If a system is structured into multiple sub-systems, a sub-system can be defined in the same way as a system is defined (i.e. as a composition of interacting elements). A hierarchy of subsystems can be limited by a "bottom-subsystem" often called device (or element in the aforementioned definition) which is not itself composed by other elements but is considered atomic. For the sake of simplicity, we consider a device as a system with only one element: itself. We then first define a system as composed by a single element and then extend the definition to a "combination of interacting" elements. Given that we use concepts from the multi-agent system research field, instead of using the word element we will use the word agent.

There is no agreement between the research communities (e.g. Multi-Agent-System, Epistemic Logic) on what the constituent of an agent are, but the same ideas revolved around for thousands of years[9]. Some relevant examples for our objective are the following.

– In [**Hintikka1962knowledge**], Hintikka describes the concept of belief (as epistemological concepts) and their relation with knowledge, and the whole Doxastic logic defines in details how beliefs can be formalized.

[9] It is interesting to notice that in [**Empiricus1990Pyrrhonism**], the author states: "The logical criterion also may be used in three senses – of the agent, or the instrument, or the "according to what"; the agent, for instance, may be a man, the instrument either sense perception or intelligence, and the "according to what" the application of the impression "according to" which the man proceeds to judge by means of one of the aforesaid instruments".

- In [**Hintikka1993Information**], Hintikka describes the concept of information (and the difference with knowledge and belief).
- In [**Santaca2016abf**], the authors defines an agent as a tuple of assertions, beliefs, and facts. A detailed discussion on facts and assertions can be found in [**facts2007stanford**] and [**assertion2007stanford**] respectively.

For our argument, as in [**Santaca2016abf**], an agent is composed by its *beliefs* (internal information considered true by the agent), and *assertions* (an exchange of information between agents). As defined in [**Hintikka1993Information**], "information is specified by specifying which alternatives concerning the reality it admits and which alternatives excludes". This means that if we consider a propositional variable (which admits the two alternatives True/False) its information is defined as believed to be True/False and not believed to be the opposite. Due to the definition of information and then with its relation to the probabilistic correlation to truth/reality, we consider information in relation to an agent's beliefs. Similarly, we consider beliefs to define the actual behavior of an agent or a system; meaning that an agent has its own beliefs on top of which it bases its behavior, the consequences of which, in turn, will update its initial beliefs. The concepts of facts and assertions as used in [**Santaca2016abf**], are not precisely discussed in [**Santaca2016abf**]. Furthermore, those epistemological concepts are difficult to define [**Gettier2012knowledge**] in general. For example, Hintikka in [**Hintikka1993Information**] states that "A purely logical definition of information is impossible". However, a detailed formalization of those concepts is out of the scope of this paper. Instead, we now define those concepts informally, inspired by the $\mathcal{ABF}$-framework defined in [**Santaca2016abf**] and, in Section 4 we detail them, and properly define them, as engineering concepts.

1. We consider information only when the intention of exchanging that information is from a sender to recipient; and we call it *assertion*.
2. Similarly, the portion of *beliefs* we consider for system engineering is the one that builds (input) or describes (output) the behavior of an agent.
3. We consider a set of axiomatic *facts* determining how the system shall be. Similarly, requirements determine which facts must hold when the system is implemented[10]. Specifically, facts/requirements describe: the functional architecture of each agent, and the physical/structural (HW/SW) architecture of each subsystem of agents and agent within a subsystem.

We give a graphical representation of (sub-)system and agent in Figure 3. The notation will be introduced in the next section but, intuitively, Figure 3 shows a system composed by two agents. The system has two ports denoted by "I" (with incoming assertions) and "O" (with outgoing assertions). The bottom part of Figure 3 details the structure of a single agent which is not decomposed into other (sub-)agents. In this case, ports translates incoming assertions into

[10] We consider facts as definitory rules[**Hintikka1993Information**] since they don't define how a real system is finally implemented but how it shall be, trough a series of requirements. Those requirements may not hold, through insecurity, in the implemented system.

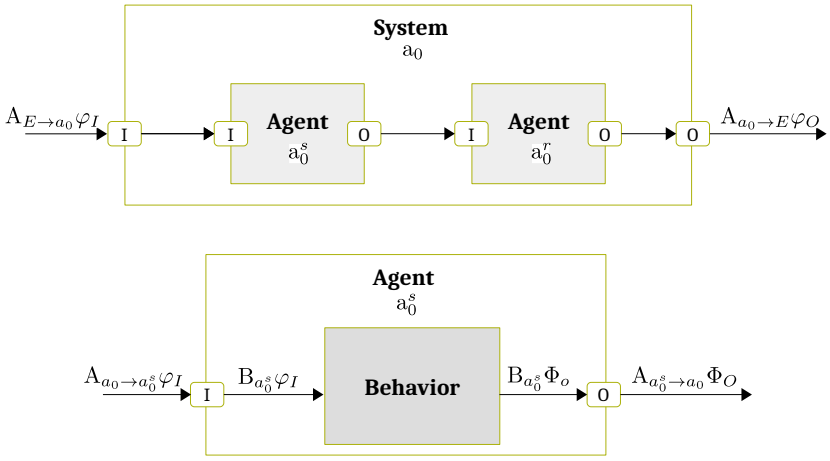

Fig. 3. Example of system and agent structure

beliefs that feeds the behavior block, and outgoing beliefs from the behavior block are then translated back into (outgoing) assertions.

3.2 Mereo-topological Reasoning

tofix: To be reviewed by Marco

Similarly to [**Santaca2016abf**], we define agents as a meronomy (an hierarchy of Part-Whole relations) over the constituents that we previously defined (assertions, beliefs, and facts), based on a standard definition of mereology, i.e. based on the definition of parthood relation between *parts*. Due to the necessity of considering different relations between parts (as we'll show afterwards) we extend the mereology to a mereo-topology [**Smith1996mereotopology**, **Varzi1994mereotopology**, **Rachavelpula2017mereotopology**], considering the relations in Table 1. For the sake of readability, we use the term *region* both to refer to a mereological part and to a topological region. Our aim is to create a meronomy (hierarchy of part-whole relations) instead of the taxonomies (categorization based on discrete sets) such as the one provided in [**NIST2020NVD**, **MITRE2020CVE**] so that we don't need to rely on a scoring system (such as the CVSS) to assign a quantitative evaluation of the cybersecurity of each entry. Instead, we want a precise calculation of the number of insecure configurations of a system to emerge from the mathematical formulation of our cybersecurity hypothesis.

A mereotopology, as defined e.g. in [**Rachavelpula2017mereotopology**], is a mathematical structure where the basic relation between regions is the reflexive and symmetric relation *Connects With*, that we use to order a universe of agents *Ag* (see in Table 1). We use the Region Connection Calculus (RCC), as defined in [**bennettLogics**, **improvingRCC**], to provide an axiomatization of the mereotopological concepts. In its broader definition, the RCC theory is composed by

RCC3	RCC5	RCC8	Terminology	Notation	Definition
			Connects with	$C(X, Y)$	reflexive and symmetric
			Disconnected from	$\neg C(X, Y)$	irreflexive or antisymmetric
			Part of	$P(X, Y)$	$\forall Z\ C(Z, X) \rightarrow C(Z, Y)$
			Overlaps	$O(X, Y)$	$\exists Z\ P(Z, X) \wedge P(Z, Y)$
●	●	●	**Equal to**	$EQ(X, Y)$	$P(X, Y) \wedge P(Y, X)$
●			**Overlaps Not Equal**	$ONE(X, Y)$	$O(X, Y) \wedge \neg EQ(X, Y)$
●	●	●	**DiscRete from**	$DR(X, Y)$	$\neg O(X, Y)$
	●	●	**Partial-Overlap**	$PO(X, Y)$	$O(X, Y) \wedge \neg P(X, Y) \wedge \neg P(Y, X)$
	●		**Proper-Part-of**	$PP(X, Y)$	$P(X, Y) \wedge \neg P(Y, X)$
	●		**Proper-Part-of-*inverse***	$PPi(X, Y)$	$P(Y, X) \wedge \neg P(X, Y)$
		●	**Externally Connected**	$EC(X, Y)$	$C(X, Y) \wedge \neg O(X, Y)$
		●	**Tangential PP**	$TPP(X, Y)$	$PP(X, Y) \wedge \exists Z\ [EC(Z, X), EC(Z, Y)]$
		●	**Tangential PPi**	$TPPi(X, Y)$	$TPP(Y, X)$
		●	**Non-Tangential PP**	$NTPP(X, Y)$	$PP(X, Y) \wedge \neg\exists Z\ [EC(Z, X), EC(Z, Y)]$
		●	**Non-Tangential PPi**	$NTPPi(X, Y)$	$NTPP(Y, X)$

Table 1. RCC3, RCC5, and RCC8 relations between regions X, Y and Z

eight axioms, and is known as RCC8. In the text, for brevity, we will often focus only on RCC5 without loss of generality. Using RCC5 instead of RCC8 prevents us from considering tangential connections between spatial regions. However, tangential connections in RCC8 can be considered as special cases of the more general spatial relations considered in RCC5.

In Table 1, we summarize the axioms of the RCC (see, e.g., [**Grutter2008rcc**]). We can now define a system over the mereotopology using the RCC calculus, as follows, where $rcc(X,Y)$ on two generic regions X,Y represents one of the possible RCC relations between X and Y. We note that all the RCC relations are symmetric with the exception of those that have an explicit (related) inverse.

Definition 1. System State – *A CPS system (or a sub-system) state is defined as the tuple* $s = \langle rcc(\mathcal{F},\mathcal{B}), rcc(\mathcal{F},\mathcal{A}), rcc(\mathcal{B},\mathcal{A})\rangle$*, where* $\mathcal{A},\mathcal{B},\mathcal{F}$*, are regions of assertions, beliefs (i.e. the beliefs generated by the behavior), and facts expressed as requirements respectively.*

As in [**Santaca2016abf**], it follows that, defining a system with a fixed number of regions, there exists an upper-bound to the number of possible configuration of a system, defined by the possible relations between the different regions. For completeness, we report in the next paragraph the calculation done by the authors in [**Santaca2016abf**].

Number of different configurations of a system The general formula to calculate the number of different types of agents is $r^{\binom{n}{k}}$, where r is the number of relations with arity k, between n different regions. Therefore, $r^{\binom{n}{k}}$ expresses the number of permutation of r relations over $\binom{n}{k}$ elements with repetitions, with $\binom{n}{k}$ being the

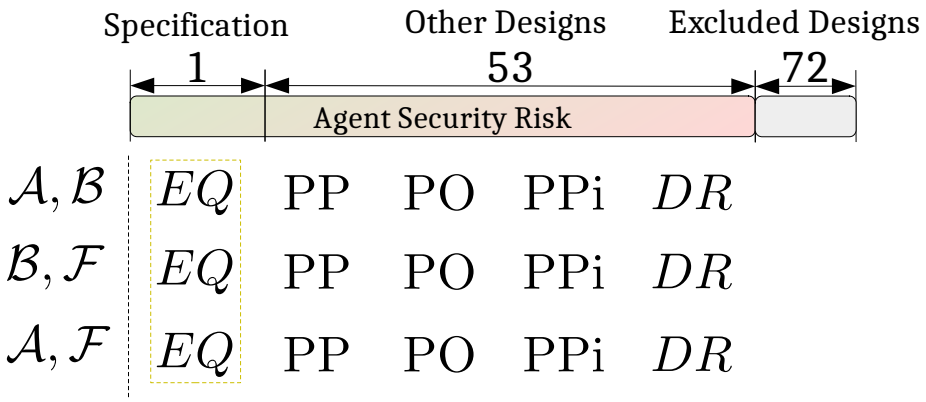

Fig. 4. Cybersecurity risk for a single agent

number of k-ary combinations of n regions. In our case, $\binom{n}{k} = 3$ since we consider 3 regions $(\mathcal{A}, \mathcal{B}, \mathcal{F})$, and all the relations considered in the RCC are binary. Hence, using RCC5 (with five different spatial relations) over three regions, we can theoretically define up to 125 different type of agents. However, only 54 of the 125 (as showed in [**improvingRCC**]) combinations are topologically correct with respect to the definition of the relations of RCC5. Generalizing to all the RCCs:

- *RCC3* — theoretical: $3^3 = 27$, correct: 15.
- *RCC5* — theoretical: $5^3 = 125$, correct: 54.
- *RCC8* — theoretical: $8^3 = 512$, correct: 193.

Hence, even if considering a different number of regions than the three $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{F}$ exponentially affects the number of theoretical agents, the application of RCC downscales that number of a factor that ranges from 1.8 to 2.5. In addition, using RCC5 we consider 3.6 times more (different) types of agents than RCC3, but using RCC8 would allow us to consider 3.5 times more different agents. In the quantitative evaluation of a single agent, depicted in Figure 4, we argue that only 1 configuration represents the nominal (expected) behavior of the agent while the other configurations are either impossible to implement or diverge from the intended nominal behavior. We note that the numbers reported here do not consider the details of the engineering process and should be considered a limit of an abstract representation of the system.

3.3 Qualitative Evaluation of Agent Space in $\mathcal{A}, \mathcal{B}, \mathcal{F}$

While a quantitative analysis reveals how many possible configurations of an agent (i.e. a system) exist w.r.t. the $\mathcal{ABF}$-framework (i.e. 54/125 in RCC5), a qualitative analysis of the different configurations describe the configurations

	DR($\mathcal{A}$,$\mathcal{B}$)	PO($\mathcal{A}$,$\mathcal{B}$)	PP($\mathcal{A}$,$\mathcal{B}$)	PPi($\mathcal{A}$,$\mathcal{B}$)	EQ($\mathcal{A}$,$\mathcal{B}$)
DR($\mathcal{B}$,$\mathcal{F}$)	T($\mathcal{A}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$) PO($\mathcal{A}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$) PO($\mathcal{A}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$)
PO($\mathcal{B}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$) PO($\mathcal{A}$,$\mathcal{F}$) PP($\mathcal{A}$,$\mathcal{F}$)	T($\mathcal{A}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$) PO($\mathcal{A}$,$\mathcal{F}$) PP($\mathcal{A}$,$\mathcal{F}$)	PO($\mathcal{A}$,$\mathcal{F}$) PPi($\mathcal{A}$,$\mathcal{F}$)	PO($\mathcal{A}$,$\mathcal{F}$)
PP($\mathcal{B}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$) PO($\mathcal{A}$,$\mathcal{F}$) PP($\mathcal{A}$,$\mathcal{F}$)	PO($\mathcal{A}$,$\mathcal{F}$) PP($\mathcal{A}$,$\mathcal{F}$)	PP($\mathcal{A}$,$\mathcal{F}$)	PO($\mathcal{A}$,$\mathcal{F}$) EQ($\mathcal{A}$,$\mathcal{F}$) PP($\mathcal{A}$,$\mathcal{F}$) PPi($\mathcal{A}$,$\mathcal{F}$)	PP($\mathcal{A}$,$\mathcal{F}$)
PPi($\mathcal{B}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$) PO($\mathcal{A}$,$\mathcal{F}$) PPi($\mathcal{A}$,$\mathcal{F}$)	T($\mathcal{A}$,$\mathcal{F}$)	PPi($\mathcal{A}$,$\mathcal{F}$)	PPi($\mathcal{A}$,$\mathcal{F}$)
EQ($\mathcal{B}$,$\mathcal{F}$)	DR($\mathcal{A}$,$\mathcal{F}$)	PO($\mathcal{A}$,$\mathcal{F}$)	PP($\mathcal{A}$,$\mathcal{F}$)	PPi($\mathcal{A}$,$\mathcal{F}$)	EQ($\mathcal{A}$,$\mathcal{F}$)

Table 2. RCC5 composition table over 3 regions. The results show that there exist 54 possible relations and the coloring anticipates the ideal risk matrix (green the secure state with low risk, red the high risk state, and a gradient of medium risk states). T($\mathcal{A}$,$\mathcal{F}$) = {DR($\mathcal{A}$,$\mathcal{F}$), PO($\mathcal{A}$,$\mathcal{F}$), PP($\mathcal{A}$,$\mathcal{F}$), PPi($\mathcal{A}$,$\mathcal{F}$), EQ($\mathcal{A}$,$\mathcal{F}$)}

allowed by the $\mathcal{ABF}$-framework, and how those configurations can be categorized. In Table 2, we provide the generic composition table of RCC5 over 3 regions instantiated over $\mathcal{A}, \mathcal{B}, \mathcal{F}$, which shows the whole state space for a single agent. The color coding of the table represents the cybersecurity risk related to a generic agent, the risk is highest on the top left corner of the matrix, lowest on the bottom right corner.

In Figure 5, the relation between facts, and assertions and beliefs (as inputs and outputs of the behavior of an agent) is illustrated. Assertions and beliefs generated by the design of a system may not be exactly aligned with what the facts mandate (i.e. what the specification mandates). The relation between facts, and assertions and beliefs can be used as a metric to determine the soundness of the design with respect to the specification.

We first analyze the relations between each pair of regions (i.e. $\mathcal{A}, \mathcal{B}, \mathcal{F}$), and then we categorize the different agents as tuple of the three regions. For the sake of simplicity, soundness is opposed to non-soundness in the following, however, with the RCC one should consider different "degrees" of non-soundness. For example, in RCC5, if we consider EQ between two regions as representing soundness, DR over the same regions represents non-soundness; while PP, PO, PPi represents the different degrees of non-soundness. A similar argument can be done for completeness.

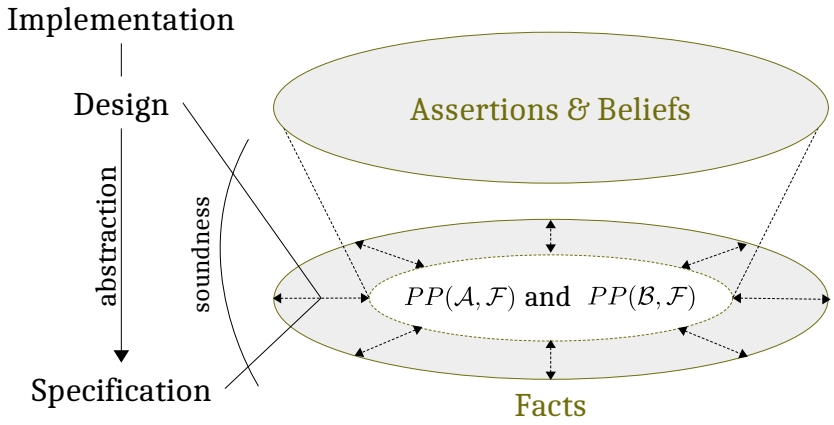

Fig. 5. Example relation between facts, and assertions and beliefs

rcc(A,B) – Collaboration In order to reason on the relation between assertions and beliefs we first need to consider that, by definition, assertions are defined as transfer of information between two agents**toremove:** , e.g., between two agents. Therefore, as depicted in Figure 3, an agent has two main categories of assertions, input and output assertions. Given an agent a and a collection of asserted predicates Φ, the input assertions are those received by a from an agent s acting as a sender, $\mathcal{A}_{s\to a}\Phi$; similarly, output assertions are sent from a to a receiver r, $\mathcal{A}_{a\to r}\Phi$. With a slight abuse of notation, in the text we simply write $\mathcal{A}_{s\to a}$ when we abstract away from the content of the assertion (i.e. Φ). We shall consider two pairs of regions:

- rcc($\mathcal{A}_{s\to a}$,$\mathcal{B}$), where the relation between input-assertions and beliefs describes the soundness of the execution of the functional architecture w.r.t. input elicitation. **toremove:** With more details, the ideal specification of the functional architecture, along with the expected inputs, defines the functional behavior of an ideal system. If all the inputs (assertions) are correctly handled in the functional specification (beliefs) the specification is complete.
- rcc($\mathcal{A}_{a\to r}$,$\mathcal{B}$), where the relation between behavior and outputs describes the completeness of the behavior defined in the specification w.r.t. the input elicitation. **toremove:** With more details, if all the outputs (assertions) of the functional architecture can be produced, the functional architecture is complete.

rcc(A,F) and rcc(B,F) – Honesty and Competence The relation of assertions and beliefs with facts determines the quality with respect to the nominal (specified) system. Given that facts define what needs to be true in the system, the relation of assertions and facts determines the degree of quality between the real information circulating in a system (or within an agent) and the one specified. Since the transfer of information through assertions generates beliefs, a dishonest agent may circulate false information, generating false beliefs. **toremove:**

We note that it is often implied that the intention behind circulating false information discerns a dishonest and an incompetent agent. However, we consider honesty related to sharing truths. The relation between beliefs and facts determines the competence (on the subjects defined by the facts) of an agent (i.e. the more competent an agent is, the more likely a belief of that agent is true).

$\mathcal{A}, \mathcal{B}, \mathcal{F}$ CyberSecurity Enumeration (CSE) The following cybersecurity requirement for a CPS specification can be summarized:

CSE-1 Proper interaction between correctly-behaving agents is defined as $EQ(\mathcal{A}_a, \mathcal{B}_a)$ for an agent a, and can be detailed as follows when multiple agents are considered.

CSE-1.1 The equality relation $EQ(\mathcal{A}_{s\to a}, \mathcal{B}_a)$ describes the intended secure behavior as: the beliefs generated by the behavior of the functional architecture shall be complete w.r.t. the specified inputs of the agent. Therefore, *the assertions received by an agent or a system shall be compliant with the expected inputs of the functional architecture.* **toremove:** For example, the inputs of the user of a SW must be sanitized to exclude deviations w.r.t. the expected inputs of the functions implemented in the SW. Another example is the type checking between allowed inputs and expected inputs.

CSE-1.2 Similarly, the equality relation $EQ(\mathcal{A}_{a\to r}, \mathcal{B}_a)$ defines that the outputs of an agent a shall be the outputs of the functional architecture.

CSE-2 The proper adherence of the data transmitted between agents with respect to requirements, is defined as $EQ(\mathcal{A}, \mathcal{F})$.

CSE-3 The proper adherence of the behavior (in terms of input and output beliefs) with respect to requirements is defined as $EQ(\mathcal{B}, \mathcal{F})$.

We note that our CSE define the properties of a secure system, and correlated weaknesses can be found in the CWE dataset. For example, CSE-1 can be seen as correlated to the weakness class of "Improper Interaction Between Multiple Correctly-Behaving Entities" defined by the CWE–435, CSE-2 with the "Insufficient Control Flow Management" defined by MITRE in the CWE–691, and CSE-3 with the "Improper Calculation" defined by MITRE in the CWE–682. All those CWE are in the top "view" of the "research concepts" in [**MITRE2020CWEresearch**], while the other classes of weaknesses do not have a direct counterpart in our hypothesis; we believe they can be sees as subclasses, but a full comparison with the CWE is out of the scope of this article.

We are now in the position to define what a secure system is (with respect to the $\mathcal{ABF}$-framework) and, based on that definition, what the cybersecurity risk is and how to quantify it in a risk matrix. The following definition holds for abstract systems defined in the $\mathcal{ABF}$-framework but will be refined for CPS afterwards in the paper.

Definition 2. Cybersecurity of a System or an Agent – *A secure system is a system where CSE-1, CSE-2, and CSE-3 holds for each agent composing the system.*

The ISO 31000 consider risk as the "effect of uncertainty on objectives" and refers both to positive and negative consequences of uncertainty. Accordingly, we consider risk as follows.

Definition 3. Risk – *The risk is the uncertainty related to the whole space of potential designs resulting from a specification in the $\mathcal{ABF}$-framework.*

The definition of Risk leads to the risk matrix in Figure 4, defined as follows.

Definition 4. Risk Matrix – *The risk matrix, as summarized in Table 2, is a function of the three relations $s = \langle rcc(\mathcal{F}, \mathcal{B}), rcc(\mathcal{F}, \mathcal{A}), rcc(\mathcal{B}, \mathcal{A}) \rangle$, where the maximum risk is defined by the DR relation between the three groups of regions, and the minimum risk by the EQ relation over the same regions. In between the two extremes, the granularity of possible intermediate configuration is defined by the calculus used (RCC5 in our case).*

While a risk matrix is often defined as a function of the likelihood and impact of attacks (based on quantitative ad-hoc estimation of how likely it is that an attacker will exploit one or more vulnerabilities and what is the magnitude of the incidents produced by this exploitation), we suggest that cybersecurity weaknesses are all equally likely to be exploited if there's a connection between the weakness and the asset that an attacker wants to impact. When the whole system is considered an asset, all weaknesses are equally likely to be exploited. Therefore, a risk matrix should capture the number of insecure configurations of a system, rather than predicating over likelihoods of weaknesses exploitation.

4 Prediction of Cybersecurity Weaknesses

Several standards mandate a secure-by-design approach in which cybersecurity shall be considered at the very early stages of the design process. **toremove:** For example,

1. DO-326A – "Airworthiness Security Process Specification" requires a cyber-security risk assessment of the design and "are the only Acceptable Means of Compliance (AMC) by FAA & EASA for aviation cyber-security airworthiness certification, as of 2019" as pointed out by SAE in [**SAE2019DO326A**].
2. NIST 800-82 [**Stouffer2011guide**] – "guide to Industrial Control System (ICS) Security".
3. J3061:2016-1 [**SAE2016J3061**] – "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems" defines "set of high-level guiding principles for Cybersecurity as it relates to cyber-physical vehicle systems" and states that "incorporate Cybersecurity into cyber-physical vehicle systems from concept phase through production, operation, service, and decommissioning".

Standards do not describe in detail how to perform a cybersecurity risk assessment and only vaguely define the overall objective, which can be summarized as to provide an understanding of the potential cybersecurity risks. **toremove:** Roughly speaking, a cybersecurity risk assessment (e.g. CORAS

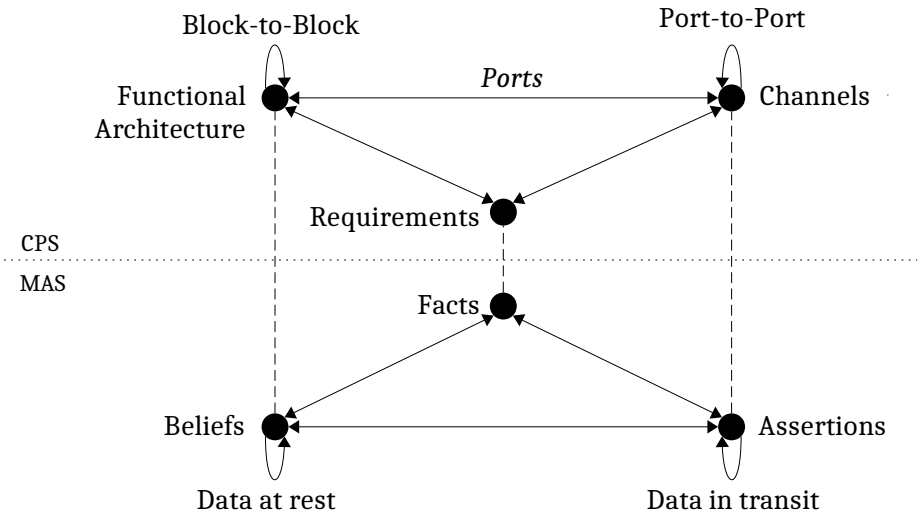

Fig. 6. Mapping Epistemological Concepts to (Cyber-Physical) Systems Engineering

[**Lund2010model**]) methodology starts from the (i) identification of assets, (ii) determines the threat scenarios (correlating vulnerabilities, threats and incidents) e.g. threat diagrams in the CORAS approach, (iii) calculates the risk as a function of impact and likelihood of threat scenarios, and (iv) finally provides treatments (mandating a partial re-design) and residual risks. All the methodologies and tools we reviewed (e.g. Threatmodeler [**Threatmodeler**], CORAS [**Lund2010model**], SECRAM [**De2015role**]) rely on the expertise of the person who performs the risk assessment for the identification of threats and for the quantitative estimation of risks. **toremove:** As Herley puts it [**Herley2016usenixvideo**], "we don't have a test for security and we don't have a metric for security" while standards require a cybersecurity risk assessment which, in turn, mandates a metric to evaluate the cybersecurity risk. In contrast, in this section, we define how to specify a CPS in the $\mathcal{ABF}$-framework and we identify a cybersecurity metric.

toremove: So far, we have reasoned on systems in the abstract, considering generic Multi-Agent System (MAS); mapping Epistemological concepts to MAS. We now map MAS to CPS and show how this mapping allows us to predict cybersecurity weaknesses in a system.

4.1 From Multi-Agent to Cyber-Physical Systems

As depicted in Figure 6, we relate MAS and CPS as follows.

– We consider a System as a hierarchy of agents. So, we map agents to *systems, sub-systems, or devices*, depending on the granularity of the design. For example, a modeler can model a specific device as a system not decomposed into sub-systems or devices (and the device is considered as an agent).

- Agents reason over beliefs (i.e. transform beliefs into other beliefs) and each *component of a CPS* (system, sub-system, or device) is composed by a *functional architecture* that transforms input-beliefs into output-beliefs.
- Components of a CPS have *ports* to exchange information with the outer environment (which may be a sub-system), similarly to agents. The transfer of information in a CPS is defined by *channels*.
- The concept of facts is related to the *requirements* that describe how the CPS shall behave and its *physical architecture*.

Input and Output Ports Since the $\mathcal{ABF}$-framework is a theory of agents, we could consider ports as agents that allow the exchange of information between a channel and another agent. However, we considered a port as a special type of agent to avoid an *infinite regress*, as described in the following. While a channel transfers information between agents, and a functional architecture elaborate information, a port is simply a connector between a channel and a functional architecture. One, however, may argue that a similar connection is needed between a channel and the port itself. While this is not excluded by the $\mathcal{ABF}$-framework, it would obviously lead to an infinite nested structure of ports. In order to avoid this infinite structure, we assume that a port doesn't require any other means to transfer information from/to a channel or from/to a functional architecture.

Definition 5. Input or Output Port – *A port forwards information from the outside of an agent's boundary to the inside (input-port) or vice versa (output-port). There exist two types of ports with the following behavior: an input-port transforms assertions from a sender s ($\mathcal{A}_{s\to a}$) to beliefs of an agent a ($\mathcal{B}_a$), while an output-port transforms beliefs into assertions.*

The *quality of a port* is determined by the *rcc* relation between the assertions received or sent and the belief, i.e. $rcc(\mathcal{A}_{s\to a}, \mathcal{B}_a)$ for the input-port or $rcc(\mathcal{A}_{r\to *}, \mathcal{B}_a)$ for the output-port (where $*$ is a wildcard for an agent). A port is, in fact, syntactic sugar to express the relation between assertions and beliefs. We note that the definition of an input/output port can be considered "secure", meaning that we implicitly formulated the requirement that a port always forwards the information without modifying it in any way. This is assumed because we define a port as if the RCC equality relation holds between the input/output assertions and the input/output beliefs. In contrast, assuming any RCC relation, between inputs and outputs of a port, which is not an equality can be considered as generating a weaknesses**toremove:** , as we describe in more details in the next paragraph.

Port Weaknesses We now apply the $\mathcal{ABF}$-framework to list all possible cybersecurity weaknesses of a port.

Theorem 1. *Port Weaknesses. From our definition of ports, it follows that there exist only the following* six *types of weaknesses, generating six types of insecure port in RCC5:*

W1) Replace port, *where assertions reach the port but are replaced with different and un-related information before passing the boundary.*
W2) Drop port, *where assertions reach the port but do not pass the boundary of the agent (i.e. do not become belief of the agent).*
W3) Insertion port, *where some new information is transferred along with the information incoming from a channel, and then sent to the intended recipient (agent).*
W4) Injection port, *where part of the incoming information is substituted with new information and transferred to the intended recipient.*
W5) Selective port, *where some information passes the port and part is either:*
W5.1) Dropped, or
W5.2) Replaced.

Proof. An input port is, in the $\mathcal{ABF}$-framework, defined secure as long as the relation between the two regions of input assertions $\mathcal{A}$ and output beliefs $\mathcal{B}$ are equal, i.e. $EQ(\mathcal{A}, \mathcal{B})$. Therefore, any other relation should result in a weakness (related to an insecurity flaw) of that input port. Using RCC5, there exist exactly other 4 different types of relations, one of which is the discrete-from (DR) relation, i.e. $DR(\mathcal{A}, \mathcal{B})$. When two regions are related by the DR relations, they have no subregion in common. Let us define a function weight $|X|$ such that, for any region X, it represents the smallest possible cardinality of a (mereo)topological base for X; where a base is a collection of regions in a (mereo)topology such that every region can be written as union of elements of that base. We distinguish between regions that are related to information and regions that don't by writing the latter as $\emptyset$ (e.g. a region corresponding to an assertion is written $A \neq \emptyset$, while a region corresponding to an empty assertion as $A = \emptyset$).

1. If $EQ(\mathcal{A}, \mathcal{B})$ then either $\mathcal{A} = \mathcal{B} = \emptyset$ (no communication) or $\mathcal{A} = \mathcal{B} \neq \emptyset$ (forward communication).
2. If $DR(\mathcal{A}, \mathcal{B})$ then $\mathcal{A} = \emptyset \oplus \mathcal{B} = \emptyset$ (we call *insert* the former and *full drop* the latter case), or $\mathcal{A} \neq \emptyset \wedge \mathcal{B} \neq \emptyset \wedge \mathcal{A} \neq \mathcal{B}$ which we call *replace* (i.e. drop and insert).
3. If $PP(\mathcal{A}, \mathcal{B})$ then $\mathcal{B}$ contains and extend $\mathcal{A}$ which we call *insertion*.
4. If $PPi(\mathcal{A}, \mathcal{B})$ then $\mathcal{A}$ contains and extend $\mathcal{B}$ which we call *drop* (or selective drop to stress the difference with the full drop).
5. If $PO(\mathcal{A}, \mathcal{B})$ then a part of the $\mathcal{A}$ is contained in the $\mathcal{B}$ which is a combination of *selective drop and generation* which we call *injection*.

Communication Channels In this work, we only consider mono-directional channels and communication but the extension to bi-directional channel can be considered as the union of two unidirectional channels. A mono-directional channel is defined by the assertions sent or received (over the channel).

We start by considering the difference between a (communication) mono-directional channel (channel from now on) and an agent, as we did for the ports, since the $\mathcal{ABF}$-framework is a logical theory of agents. In fact, if a channel were considered an agent (channel-agent) then the question would be how an agent would transfer its assertions to the channel-agent. If the channel between the

agent and the channel-agent is again an agent, we would generate an *infinite regress*. Therefore, we do allow channel-agents but we assume a finite depth (of detail) for a channel, where there exists a bottom-channel which is not an agent. For now, we do not constrain a channel-agent in any way so there is no difference between a channel agent and agent. Therefore, we consider channels to be bottom-channels, defined as agents with the pre-defined behavior (i.e. defined in an axiomatic way) of forwarding any input-assertion as output-assertion, without modifying it[11].

Definition 6. Mono-directional Channel (bottom-channel) – *A mono-directional channel between two agents ($s \rightarrow r$) is defined as an agent whose behavior is (dogmatically) defined as: to forward any assertion received from s over an input-port, to the output-port where r is listening to.*

The *quality of a mono-directional* channel is defined as the *rcc* relation between the assertions made by the sender and the ones received by the receiver, i.e. $rcc(\mathcal{A}_s, \mathcal{A}_r)$.

Channel Weaknesses Given that a mono-directional bottom-channel is assumed to be perfectly forwarding any assertion (as we assumed for ports) from its input-port to its output-port, there is no insecure behavior but only the combination of the weaknesses of the input and output port; therefore there exist $(7^2) - 1 = 48$ theoretical configurations (7^2 because there are 6 insecure types of port, plus 1 secure type, on both input and output side; and we exclude the configuration with 2 secure types as input and output, -1); where only 44 are possible. For the sake of readability, we report 6 examples in the following but the proof by exhaustion over all the possible cases is straightforward.

W6) *Secure output port* and *input drop port.*
W7) *Secure output port* and *input insertion port.*
W8) *Output drop port* and *input drop port.*
W9) *Output drop port* and *input insertion port.*
W10) *Output drop port* and *input secure port.*
W11) *Output injection port* and *input secure port.*

Cybersecurity Weaknesses – The RIDI-Hypothesis It is important to note that all the results of the application of the $\mathcal{ABF}$-framework to channels (the analysis of the RCC relations between output and input assertions of an agent) lead to the same results of the analysis of a pair of an input and output port. So far we have considered information generated by a port P_I and then sent through a channel C to another (recipient) port P_O. In this scenario, where ports and channels are atomic (otherwise raising infinite regress), we can only consider the relations between ports and channel; considering both input-port to channel and channel to output-port. In fact, the weaknesses of a channel are defined in terms of the

[11] Nothing prevents us from introducing additional constraints to the channel as storing assertions that are transferred over the channel, or filter out some input-assertions.

weaknesses of ports. The same result can be obtained by analyzing the relation between the outputs of a functional block and the inputs of another functional block, where functional blocks are constituents of the functional architecture as described afterwards. In order to define a functional block without encountering an infinitely recursive definition, we must reach the same conclusions as for the channel. Therefore, describing the information as flowing over a channel or in a functional block is purely syntactic sugar.

We can summarize these results by saying that the relations between assertions and beliefs, output assertions of an agent and input assertions of another agent, or output beliefs of a functional block and input beliefs of another block can *only* be affected by the following weaknesses: replace, drop, injection, insertion, selective drop, and selective drop + insertion. We call this the RIDI-Hypothesis, being the four main categories of weaknesses: Replace, Insertion, Drop, Injection. We can, then, deduce the following cybersecurity properties to mitigate cybersecurity weaknesses of a port or a channel (between ports, functional blocks, or both).

- *Order-preserving* – it shall be known if information is *replaced*.
- *Availability* – it shall be known if information is *dropped* or *selectively dropped*.
- *Integrity* – it shall be known if information is *injected*.
- *Authentication* – it shall be known if information is *inserted*.

Functional Architecture A functional architecture takes information as input-beliefs and transforms the information into output-beliefs. Those transformations occur within the functional architecture, where functional blocks transform beliefs into other beliefs. Similarly to channels, we could consider a functional block as a functional architecture occurring in an infinite regress. Therefore, we consider functional blocks as executing an abstract undefined behavior, of which we only observe the inputs and the resulting outputs (beliefs).

Definition 7. Functional Block and Architecture – *A functional block of an agent takes beliefs as inputs (input-beliefs) and returns output-beliefs. A functional architecture is an interconnected system of functional blocks.*

The quality of a functional block cannot be determined by the difference between its inputs and outputs (as we did for ports and channels), because the behavior of a functional block cannot be determined in general; since any functional block will have its own purpose based on functional requirements. Therefore, while the semantics of a port is determined by the relation between assertions and beliefs, the semantics of a functional block is determined by the relation between facts/requirements and I/O beliefs. In other words, a functional block is a generic agent with no pre-defined general behavior (while ports and channels have a pre-defined behavior). In the following, for the sake of simplicity, we use the generic region $\mathcal{B}$ to refer to the behavior (i.e. the beliefs generated by the behavior).

W50) $PO(\mathcal{B}, \mathcal{F})$ the component has a Byzantine behavior where occasionally outputs the expected output given the correct inputs. Not all the inputs are

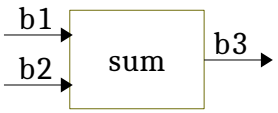

Fig. 7. Example of a Functional Block

handled properly, nor all the expected outputs are always generated when correct inputs are given.

W51) $PP(\mathcal{B}, \mathcal{F})$ part of the expected outputs are not generated in response to the correct inputs

W52) $PPi(\mathcal{B}, \mathcal{F})$ the components correctly perform the expected behavior when the correct inputs are provided but is subject to input injections

W53) $DR(\mathcal{B}, \mathcal{F})$ the component never performs the expected behavior (e.g. physical damage)

Requirements as Facts During the specification phase, for any agent, channel, port, functional block and architecture, there may exist a requirement (fact) predicating over them. In other words, any requirement is defined as a fact since they must be true in any design or implementation. As we stated in Section 4 and depicted in Figure 5, facts are definitory rules that define how the system shall behave (by specification), while reality may be shown to be insecure (i.e. diverging from the expected behavior).

As an example, considering a functional block that performs the summation of two inputs, as in Figure 7, defined by the requirement $r := b_3 = b_1 + b_2$ for any b_1 and b_2. The possible relations between the beliefs generated by the behavior of the functional block and the requirements (i.e. $rcc(\mathcal{B}, \mathcal{F})$ is determined by the relations between the I/O beliefs $sum(b_1, b_2) = b_3$ and the requirement $sum(b_1, b_2) = b_1 + b_2$, as follows.

- $EQ(\mathcal{B}, \mathcal{F}) = EQ(\mathcal{B}_3, \mathcal{B}_{1+2})$, where $\mathcal{B}_3$ represents the region of the outputs of the functional block *sum* while the region $\mathcal{B}_{1+2}$ represents the expected outputs of an ideal implementation of the requirement r. Therefore, the functional block correctly implements the requirements.
- $DR(\mathcal{B}, \mathcal{F}) = DR(\mathcal{B}_3, \mathcal{B}_{1+2})$, the function block never implements the requirements.
- $PP(\mathcal{B}, \mathcal{F}) = PP(\mathcal{B}_3, \mathcal{B}_{1+2})$, there exist some inputs for which the output is incorrect.
- $PPi(\mathcal{B}, \mathcal{F}) = PPi(\mathcal{B}_3, \mathcal{B}_{1+2})$, there exist some outputs that are not the result of the summation of the inputs but given the correct inputs the function always outputs correctly.
- $PO(\mathcal{B}, \mathcal{F}) = PO(\mathcal{B}_3, \mathcal{B}_{1+2})$, the component has a Byzantine behavior where occasionally outputs the expected output given the correct inputs. Not all the inputs are handled properly, nor all the expected outputs are always generated when correct inputs are given.

Assertions and Facts The whole reasoning on the relation between beliefs and facts can be duplicated for the relation between assertions and facts; we cannot appreciate the difference at this level of abstraction. If the functional architecture would be extended to capture the semantics (i.e. the logic) of the communication and cybersecurity protocols, with the relation between assertions and facts we would compare protocol logics with the requirements. We won't consider the verification of the functional architecture and protocol logic in this paper since we focus on the architecture specification step of the engineering process without going into the design of the behavior of agents. **toremove:** We can give an example but research is needed to apply our hypothesis to the behavior of agents.

If we consider a functional block that generates nonces (i.e. number used only once), the semantics of this block must define a procedure such that a freshness requirement holds. By the RIDI-Hypothesis we have that the following weaknesses may occur:

- Replace: the output-nonces of the block are replaced with non-fresh numbers.
- Insert: the output-nonces are concatenated with other numbers, resulting in non-fresh nonces. For example, the nonces 110 and 111 are modified by inserting a 1 and a 0 at the beginning and end respectively, resulting in the two identical (and the non-fresh) nonces 1110 and 1110.
- Delete: part of output-nonce is dropped. For example, the last bit of 1101 and 1100 is dropped, resulting in the two equal and non-fresh nonces: 110 and 110.
- Inject: substitution at bit level results in non-fresh nonces. For example, 110 and 111 becomes equal by flipping either the last 0 of the former or the last 1 of the latter.

We summarize our results by categorizing the weaknesses predicted by our hypothesis into: data-flow-related and functionality-related weaknesses; as in Table 3. Functionality-related weaknesses can be seen as a general formulation of our hypothesis while data-flow-related weaknesses as an application of our hypothesis to components with defined behavior/requirements.

4.2 Security and Insecurity of a System

Hypothesis 1 System Security Design – *A system security design (in the $\mathcal{ABF}$-framework) is given by a precise system specification over the physical and functional architectures that uniquely defines the design built on top of those requirements.*

Hypothesis 2 System Insecurity Design – *If, given a system specification as a collection of requirements, there exist a non-unique design with respect to those requirements, the number of equivalent designs (that fulfills the requirements) quantitatively defines the magnitude of insecurity of a system design.*

Based on these hypothesis, we can formulate the concepts of security and insecurity (in the $\mathcal{ABF}$-framework) as mathematical equations. Let us consider

RCC5	Quantity: Data Flow	Quality: Requirements Adherence
EQ	Expected/Nominal	Expected/Nominal
DR	Drops all the inputs and inserts new malicious data	The component never performs/carries the expected behavior/information
PP	Selectively drops inputs	part of the expected outputs are not generated in response to the correct inputs
PPi	Forwards all the inputs but crafts and inserts new malicious data	The components correctly performs/carries the expected behavior/information when the correct inputs are provided but is subject to input injections
PO	Selectively drops inputs and inserts new data	Byzantine behavior - occasionally outputs the expected output given the correct inputs. Not all the inputs are handled properly, nor all the expected outputs are always generated when correct inputs are given

Table 3. Weaknesses Categorization

a CPS S, represented as a graph $G = \langle V, E \rangle$ where V represents the set of functional blocks and ports of S, and $E \subseteq V \times V$ is the set of pairs representing the channels and connections (data flows) between functional blocks. We define $R \subseteq V \times F$, where F is the set of all the requirements of S, and extend G as $G' = \langle V', E' \rangle$ with $E' = E \cup R$ and $V' = V \cup F$. Let $\pi : E' \to RCC$ (where RCC is the set of relations in the RCC) be the total function associating an RCC relation to each edge in G', and Π be the set of all different permutations of RCC relations over E' (i.e. $\Pi = \{\langle \pi(e_0) = EQ, \ldots, \pi(e_n) = EQ \rangle, \ldots, \langle \pi(e_0) = DR, \ldots, \pi(e_n) = DR \rangle\}$ where $e_i \in E'$ for $0 \leq i \leq n$ and $|E'| = n$). If $\sigma : \Pi \to \{0, 1\}$ is an evaluation function such that $\sigma(p) = 1$ (where $p \in \Pi$) iff the input configuration is satisfiable with respect to the logical theory defining the algebraic structure (mereotopology) and constraints of the calculus RCC (otherwise σ returns 0), we can define

$$I = \sum_{p \in (\Pi \setminus \pi_{eq})} \sigma(p)$$

where $\pi_{eq} \in \Pi$ is the output of the function π that associates only EQ relations to any $e \in E'$, and $I \in \mathbb{N}$ represents all the insecurity configurations of the CPS S where, at least, one of the RCC relations isn't EQ. In other words, we consider π_{eq} as the only secure configuration.

4.3 Cybersecurity Risk Assessment

In order to test our hypothesis we implemented a tool-chain (open-source under the AGPLv3 license and available at [**v-research2020cybersecurity-anonymous**])

for the identification of weaknesses and the precise calculation of potential insecure configurations. The engineering of the $\mathcal{ABF}$-framework for CPS is summarized in the UML Class diagram in Figure 12 (in appendix).

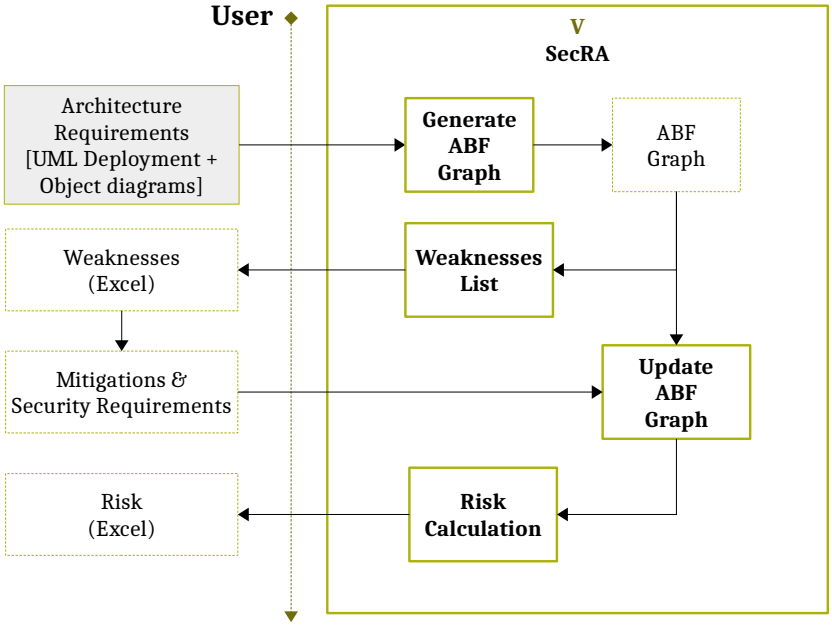

Fig. 8. Cybersecurity Risk Assessment Tool

As depicted in Figure 8, the cybersecurity risk assessment process starts with the definition of the use cases and architectural requirements. In our process, the specification is manually translated into a UML design where:

- a *deployment diagram* describes the *physical architecture*. Each agent is defined as an UML node with (physical) ports, and agent' ports are connected via UML information flow connectors, representing the physical channel.
- a *functional architecture* is linked to each agent in the deployment diagram and is defined by an *object diagram*. The object diagram is composed by instances of functional blocks, connected via information flow connectors.
- the connection between the two diagrams is implemented by "sockets", functional blocks connected to a physical port.

The tool generates a graph-like internal structure which represents the specification (called $\mathcal{ABF}$-graph). The $\mathcal{ABF}$-graph defines the system as a number of regions of assertions, beliefs, and facts. Those regions are connected by a generic relation which is evaluated as follows (according to the formula in Section 4.2). The graph is translated into a logical formula that represents the specification in

the $\mathcal{ABF}$-framework and, along with the axiomatization of the RCC5 calculus, is given as input to the Z3 SMT solver. The solver identifies all possible configurations of the system and, in turn, identifies all potential weaknesses. The internal structures of the tool can be viewed as PDF and the results are reported into an spreadsheet file. The spreadsheet file also reports the total number of configurations as indicating the cybersecurity risk associated to the specification. A user can change the status of each weakness in the spreadsheet file, from the initial (default) status (open) to "mitigated". As a consequence, the risk is re-calculated on-the-fly, i.e. without the need of running the tool again, based on annotations and formulas in the spreadsheet file. In our approach, cybersecurity requirements are not imposed by the specification but are automatically extracted by our tool as mitigations to potential weaknesses. Those weaknesses are related to the different insecure configurations of the specified system.

Case Studies We report the results of the evaluation of a water level reader (sensor ad-hoc example). As in Figure 9, we defined 2 agents: sensorInTank and

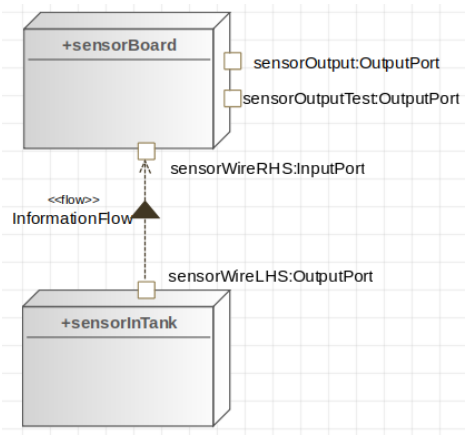

Fig. 9. Water level reader deployment diagram

sensorBoard, which represent the physical reader that needs to be placed in a tank, and the board that interprets the readings and outputs them as signals. The two components are connected by a wire. In Figure 10, we report the functional architecture that receives the incoming communications from the sensor in the tank, and communicates them encrypted. The results, summarized in Figure 11, reports 16777216 possible scenarios in which at least one component diverge from the specification.

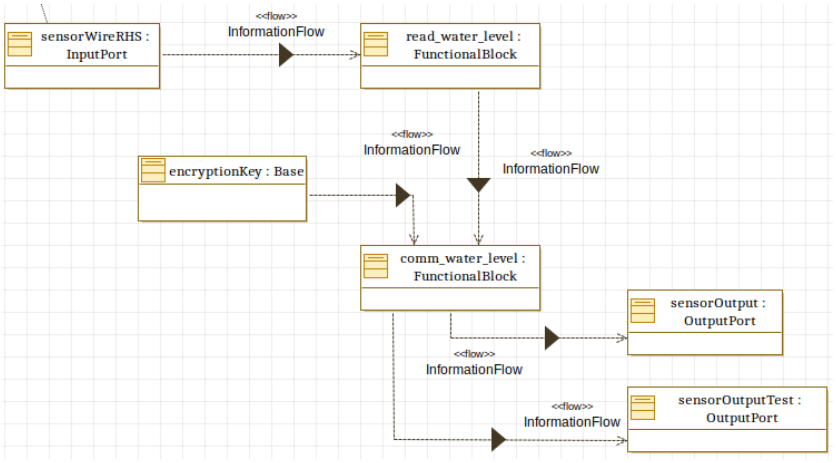

Fig. 10. Sensor Board Object Diagram

RISK 16777216

The total risk is the total number of insecure configurations of the system

B	C	D	E	F
Agent	**Component**	**Comp. Type**	**Weakness**	**Status**
sensorBoard	sensorWireRHS	inputport	selectively drops inputs and inserts new malicious data	open
root	sensorWireLHS2sensorWireRHS	channel	selectively drops inputs and inserts new malicious data	open
sensorInTank	sensorWireLHS	outputport	selectively drops inputs and inserts new malicious data	open
sensorInTank	sensorWireLHS	outputsocket	the component has a Byzantine behavior where occasionally outputs the expected output given the correct inputs. Not all the inputs are handled properly, nor all the expected outputs are always generated when correct inputs are given.	open

Fig. 11. Partial View of the results in the spreadsheet file

5 Conclusion and Future Works

We proposed an hypothesis for a foundational theory on security, arguing that cybersecurity-related issues are not linked to the maliciousness of an agent but to the vagueness in the design processes. We also provided a prototype tool for the quantitative estimation of the cybersecurity risk, based on a UML model of the system under assessment. The cybersecurity verification and test-case generation will be the focus of our next steps.

The Class Diagram for the Engineering of the $\mathcal{ABF}$-framework is reported in Figure 12. A *specification* of a CPS is viewed as an aggregation of *architectures* which can describe the functional or physical requirements. The physical components of the architecture are input/output *ports* and *channels* (aggregations of pairs of ports) while *functional blocks* are the only constituents of the functional architecture. All of the classes are abstract except input/output ports and functional blocks. Therefore, agents (which represents sub-systems or components) are composed by ports and functional blocks, as an aggregation of architectures.

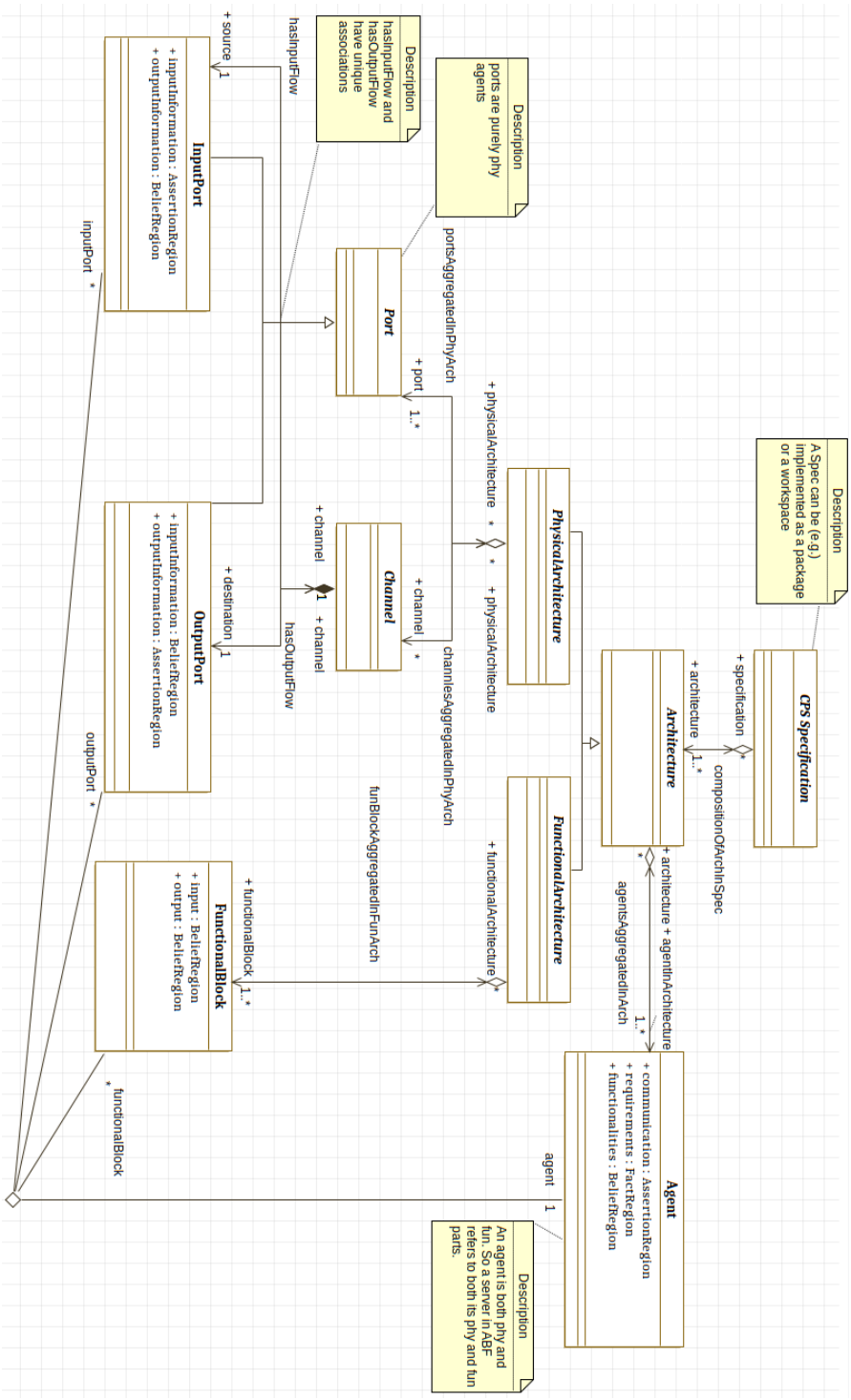

Fig. 12. $\mathcal{ABF}$-framework for CPS Design – Class Diagram