Identifying Implicit Component Interactions in Distributed Cyber-Physical Systems

Jason Jaskolka
Center for International Security and Cooperation
Stanford University
jaskolka@stanford.edu

John Villasenor
Department of Electrical Engineering
University of California, Los Angeles
villa@ee.ucla.edu

Abstract

Modern distributed systems and networks, like those found in cyber-physical system domains such as critical infrastructures, contain many complex interactions among their constituent software and/or hardware components. Despite extensive testing of individual components, security vulnerabilities resulting from unintended and unforeseen component interactions (so-called implicit interactions*) often remain undetected. This paper presents a method for identifying the existence of implicit interactions in designs of distributed cyber-physical systems using the algebraic modeling framework known as Communicating Concurrent Kleene Algebra (*C^2KA*). Experimental results verifying the applicability of* C^2KA *for identifying dependencies in system designs that would otherwise be very hard to find are also presented. More broadly, this research aims to advance the specification, design, and implementation of distributed cyber-physical systems with improved cybersecurity assurance by providing a new way of thinking about the problem of implicit interactions through the application of formal methods.*

1. Introduction and motivation

Cyber-physical system domains, such as those found in critical infrastructures and other safety-critical industries, like the aerospace and automotive industry, are typically designed as large, complex, and distributed networks consisting of many interacting software and hardware components. Although individual system components are often subject to rigorous testing and verification, there is often no guarantee that the overall system will always function as expected once the components are combined due to their many complex interactions. Because of the size and complexity of today's distributed systems, a significant number of these interactions may not be expected or foreseen by the system designer (i.e., they are *implicit interactions*).

In large and complex systems, a significant number of issues can result from interactions among system components that are satisfying their requirements [44]. As such, implicit interactions do not necessarily represent errors in the system. For example, the presence of an implicit interaction in a given system does not necessarily mean that it fails to perform the functionality that it was intended to perform. Rather, the existence of implicit interactions can indicate unforeseen flaws in the system design allowing for the potential for additional, and often unwanted, system behaviors to manifest during the operation of the system. Additionally, or alternatively, such interactions can be symptoms of intentionally compromised software and/or hardware specifically designed to remain undetected in tests formulated to identify accidental design flaws in the individual components. Such compromised software and/or hardware can be exploited to mount a cyber-attack at a later time. Therefore, this notion of implicit interactions must be carefully managed, particularly at early stages of system development, to have systems that operate as intended, and that are resistant to cyber-attacks.

This paper presents a formal methods-based approach for identifying the presence of implicit interactions in the designs of distributed cyber-physical systems. The proposed approach intends to provide a new way of thinking about this problem by applying formal methods through the specification and analysis of the communication among system components using the algebraic modeling framework called Communicating Concurrent Kleene Algebra (C^2KA) [34], [35]. The techniques presented here address an increasingly important need, particularly in light of the growth in complexity of distributed systems and networks in critical infrastructures and other sectors, and the shortcomings of formal methods for determining whether such systems are protected from cyber-threats [6]. Additionally, they help to identify vulnerabilities in the designs of important existing system components, allowing for better assessment of the risks being taken by using such components in the development of systems. More generally, the objective of this paper is to provide a rigorous and practical approach for advancing cybersecurity assurance of distributed cyber-physical systems at early stages in their development.

The rest of this paper is organized as follows. Section 2 considers the related work and differentiates our contributions from the existing literature. Section 3 outlines an illustrative example that will be used to demonstrate the proposed approach throughout the remainder of this paper. Section 4 outlines the proposed approach for identifying implicit interactions and presents our experimental results. Section 5 discusses the proposed approach and comments on its scalability. Finally, Section 6 gives concluding remarks and highlights our future work.

URI: http://hdl.handle.net/10125/41886
ISBN: 978-0-9981331-0-2

2. Survey of the literature

In this section, we survey the literature to discuss the large body of existing work in the area of analyzing component interactions in complex systems. We compare and contrast existing formalisms and approaches for modeling and analyzing component interactions and we differentiate our contributions from this existing work.

2.1. Formalisms for modeling complex distributed systems

The development of formalisms for modeling complex distributed systems has received much attention in the past. Many of the proposed formalisms aim to capture the communication, concurrency, and dynamics of the components that comprise a given system. These existing formalisms include the Actor Model (e.g., [26], [20], [5], [10]), as well as process algebras (e.g., CCS [48], CSP [27], ACP [7], and π-calculus [50]), architectural formal modeling languages (e.g., AADL [14], EAST-ADL [11], and SysML [17]), labelled transition systems [38], Petri nets [54], synchronization trees [48], event structures [70], action algebras [40], [43], [55], and Concurrent Kleene Algebra (CKA) [28].

While each of the above mentioned formalisms and languages have their merits, in this paper, we elect to use Communicating Concurrent Kleene Algebra (C^2KA) [34], [35] for the system modeling and specification in the proposed approach. C^2KA provides a hybrid view of communication and concurrency encompassing the characteristics of both state-based and event-based models which is desirable for specifying systems at various levels of abstraction. By contrast, other formalisms for capturing the communication, concurrency, and dynamics of complex distributed systems do not directly, if at all, provide such a view. Furthermore, other formalisms do not directly deal with describing how the behaviors of components within a system are influenced by stimuli. When considering open systems, stimuli are required in order to initiate behaviors. Many other formalisms, such as CKA and process algebras, deal primarily with closed systems where there is no external influence on the behaviors of system components and they do not directly, if at all, consider behaviors in open systems.

2.2. Interference/Non-interference

The problem of implicit component interactions is closely related to the notion of non-interference [19]. Often when discussing non-interference, a system is modeled as a machine with inputs and outputs, each classified as either low-level or high-level. A system has the non-interference property if and only if any sequence of low-level inputs will produce the same low-level outputs, regardless of what the high-level inputs are, meaning that the low-level components are not influenced by the high-level components of the system. The study of non-interference emerged from the need to understand why particular undesirable interactions among system components were possible [58].

Numerous approaches for ensuring the satisfaction of non-interference properties have been proposed. Goguen and Meseguer [19] defined the existence of undesirable component interactions through non-interference properties in security policies. Volpano and Smith [68] described non-interference through typing where a system contains interference if it cannot be correctly typed, and Ryan [57] and Lowe [46] described non-interference within a process algebraic framework. More recently, there has been work in addressing issues of intransitive non-interference [66], as well as in examining non-interference over system architectures [8].

According to [24], there is a cost for characterizing system cybersecurity in terms of non-interference assertions. This is due to the fact that a relatively complicated induction is necessary to verify that a non-interference policy is satisfied. Furthermore, non-interference approaches often attempt to classify system components according to two security levels: high and low [25]. However, it is rarely the case that real systems only have two security levels, which leads to a fundamental restriction of the use of non-interference properties. We aim to develop an approach concerned primarily with studying the influence of systems and components on one another through their communication, and that offers a level of abstraction that does not require any rigid classification of system components.

2.3. Information flow analysis

The study of information flow has been considered one of the primary approaches for studying the interactions of components in complex distributed systems and networks [16]. For example, Denning [12] investigated a lattice structure derived from security classes to guarantee secure systems and information. McDermid and Shi [47] and Shaffer et al. [61] focussed on identifying system vulnerabilities at the implementation level through static analysis techniques. A variety of other approaches captured information flow security requirements using various formalisms including state machines (e.g., [62]), Petri nets (e.g., [67]), process algebras (e.g., [15]), typing systems (e.g., [23], [30], [39], [69]), and axiom systems (e.g., [4], [59]).

Models of information flow attempt to describe all possible ways of comprising information at fine-grained views of a system. With this view, the information is recognized as low as the bit level. Information flow analysis also typically occurs at later stages in software development, such as the implementation stage. For instance, typing systems are able to analyze information flows within program code, but not at an earlier development stage. Instead, we aim to develop an approach that can identify implicit component interactions at much earlier stages in system development. A similar idea has been carried out by Alghathbar et al. [2]. This work proposed FlowUML which aimed at validating information flow policies in UML sequence diagrams in an effort to detect information flow violations at an earlier stage of system development.

2.4. Other related work

In addition to the above, other approaches aimed at investigating the interactions of system components have also been proposed using various formalisms, methods, and techniques. For example, the problem of undesired component interactions has been well-documented in the area of hazard analysis and system safety with the development of procedures for identifying potential failure modes and events in complex systems (e.g., [44], [64]). There is also a wealth of literature describing the kinds of vulnerabilities, threats, and challenges that exist in distributed cyber-physical systems, including those found in critical infrastructure sectors. Many studies are focussed on capturing the interdependencies among critical infrastructure sectors and studying the effects of (cascading) failures using modeling and simulation approaches (e.g., [13], [53], [56]). Other existing research looks to perform risk assessments using a variety of techniques including those based on network analysis and fault trees (e.g., [45]), anomaly detection (e.g., [31]), and through the examination of access control mechanisms (e.g., [37]). However, probabilistic risk assessments place an emphasis on identifying and dealing with failure events, with design errors only being considered indirectly through the probability of the failure event. Issues resulting from unwanted or unexpected component interactions and systemic factors are typically not considered.

In contrast, a large proportion of existing work has aimed to provide formal analysis and verification of concurrent systems (e.g., [42], [18], [3], [63]), as well as the formal verification of dynamic and parametrized systems and networks (e.g., [9], [1], [51], [52]), as means of providing assurances that systems operate as expected as they continue to grow in size and complexity. Similarly, recent work aiming to formally identify risks and provide solutions for safely managing the complexity in the design and operation of flight-critical systems using category theory and the idea of *operads* has been proposed [60]. However, a significant proportion of this work does not explicitly focus on addressing the issue of implicit component interactions at the design stage of system development. By contrast, in the present paper we provide an approach that examines the component interactions of distributed cyber-physical systems by analyzing the potential communication paths arising from the system design and specification using an alternative abstract algebraic framework.

Although many formalisms and approaches already exist which aim to address and discuss issues related to implicit component interactions, we propose an alternative approach meant to aid system designers in formally and systematically assessing their designs by helping to identify potential security vulnerabilities and risks at early stages of system development. We aim to provide a different and complementary perspective for studying the interactions of system components than what is offered by existing formalisms and approaches.

3. Illustrative example: Manufacturing cell control system

Distributed control systems are an important aspect nearly every cyber-physical system, including critical infrastructures, as well as the aerospace and automotive industries. To demonstrate the proposed methodology, we consider an illustrative manufacturing cell control system, adapted from [21], consisting of four agents[1] (components): *Control Agent* C, *Storage Agent* S, *Handling Agent* H, and *Processing Agent* P. The expected behavior and operation of the manufacturing cell control system can be visualized as shown in the message passing diagram in Figure 1.

When the system is ready to begin manufacturing, a *start* event is triggered. C begins the manufacturing process by sending a *load* request to S which responds by entering its loaded behavior (FULL). When the loading is complete, S broadcasts a *loaded* message. C responds by transitioning to its preparation behavior (PREP) and by sending a *prepare* request. H responds by transitioning to its moving behavior (MOVE) and by sending an *unload* request to S which responds by entering its unloaded behavior (EMPTY). After unloading, S broadcasts an *unloaded* message that causes C to transition to its initializing behavior (INIT) and to issue a *setup* event. P responds by entering its setup behavior (SET) and by sending a *ready* message which causes H to transition to its waiting behavior (WAIT) and to send a *process* event. Both P and C respond by moving to their working (WORK) and processing (PROC) behaviors, respectively. Once P is finished working, it issues a *processed* event that causes C to return to its idle behavior (IDLE). Similarly, when C is finished processing, it issues a *done* message causing P to return to its standby behavior (STBY). At this point, the control system awaits another *start* event to begin the manufacturing process again.

This illustrative manufacturing cell control system will be used as a running example to show to how to use the algebraic modeling framework C^2KA to specify and analyze systems to identify implicit interactions. While this example is presented in the context of manufacturing, the analogous message passing and dependencies are found in nearly all distributed cyber-physical systems.

4. The proposed approach

In this section, we articulate the proposed approach for identifying implicit component interactions in distributed cyber-physical systems. First, we demonstrate how to model distributed systems using the algebraic modeling framework known as Communicating Concurrent Kleene Algebra (C^2KA). Then, we provide a formulation of the existence of implicit interactions and show how to identify implicit interactions in a system modeled using C^2KA. Finally, we present our experimental results of the analysis of the illustrative manufacturing cell control system described in Section 3.

1. The term *agent* refers to any system, component, or process whose behavior consists of discrete actions [49].

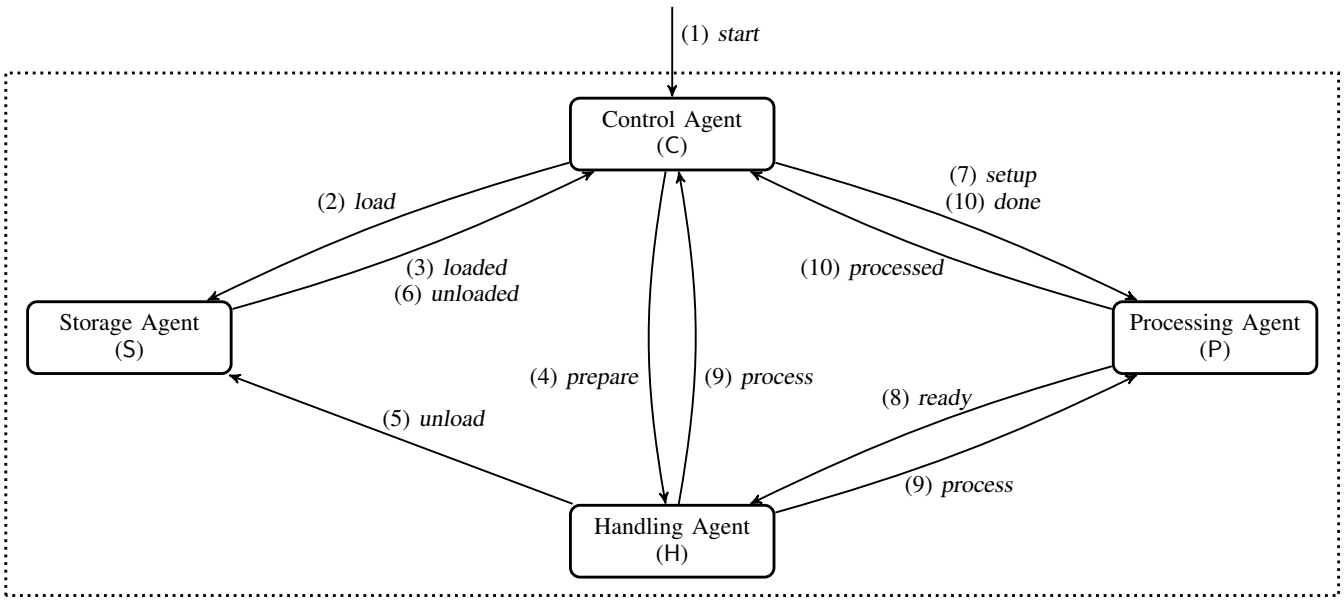

Figure 1. Message passing diagram depicting the behavior and operation of the manufacturing cell control system

4.1. Modeling systems using C^{2}KA

Communicating Concurrent Kleene Algebra (C^{2}KA) [34], [35] is an algebraic framework for capturing the concurrent and communicating behavior of agents in a distributed system. In essence, a C^{2}KA is a mathematical system that describes how a stimulus structure $\mathcal{S}$ and a concurrent Kleene algebra (CKA) $\mathcal{K}$ mutually act upon one another in order to characterize the response invoked by a stimulus on an agent behavior as a next behavior mapping (denoted by $\circ$) and a next stimulus mapping (denoted by λ). Formally, a C^{2}KA is defined as shown in Definition 1.

***Definition 1* (C^{2}KA – *e.g., [35]*).** A *Communicating Concurrent Kleene Algebra* (C^{2}KA) is a system $\big(\mathcal{S}, \mathcal{K}\big)$, where $\mathcal{S} = \big(S, \oplus, \odot, \mathfrak{d}, \mathfrak{n}\big)$ is a stimulus structure and $\mathcal{K} = \big(K, +, *, ;, \circledast, \odot, 0, 1\big)$ is a CKA such that $\big({}_{\mathcal{S}}K, +\big)$ is a unitary and zero-preserving *left $\mathcal{S}$-semimodule* with mapping $\circ : S \times K \to K$ and $\big(S_{\mathcal{K}}, \oplus\big)$ is a unitary and zero-preserving *right $\mathcal{K}$-semimodule* with mapping $\lambda : S \times K \to S$, and where the following axioms are satisfied for all $a, b, c \in K$ and $s, t \in S$:

1) $s \circ (a\,;b) = (s \circ a)\,;\big(\lambda(s, a) \circ b\big)$
2) $a \leq_{\mathcal{K}} c \;\vee\; b = 1 \;\vee\; (s \circ a)\,;\big(\lambda(s, c) \circ b\big) = 0$
3) $\lambda(s \odot t, a) = \lambda\big(s, (t \circ a)\big) \odot \lambda(t, a)$
4) $s = \mathfrak{d} \;\vee\; s \circ 1 = 1$
5) $a = 0 \;\vee\; \lambda(\mathfrak{n}, a) = \mathfrak{n}$

We refer the reader to [34] and [35] for the full detailed description and presentation of C^{2}KA.

We chose to use C^{2}KA as the framework for specification and analysis since it provides the capability to model open systems with the notion of external stimuli coming from outside the boundaries of the system being considered. C^{2}KA offers a hybrid view of communication and concurrency for specifying systems at multiple levels of abstraction that is not provided by other existing formalisms. Even with a formalism such as CKA [28], which can be seen as a hybrid model for concurrency, the notion of communication is not directly captured. Communication can only be perceived when programs are given in terms of the dependencies of shared events, thereby requiring the instantiation of a low-level model of programs and traces for CKA in order to define any sort of communication [29]. Instead, we wanted a way in which communication can be specified using CKA without the need to articulate the state-based system of each action (i.e., at a convenient abstract level). Furthermore, by capturing the influence of stimuli on behaviors within a system, C^{2}KA is also able to capture the dynamic behavior of distributed systems and networks. Furthermore, the use of C^{2}KA for modeling and specifying complex distributed systems makes it straightforward to ascertain the potential communication paths that exist in a given system. Consequently, it is easy to develop a perception of the overall topology of a given system with respect to its specification which allows for the abstraction of components of the overall system behavior. Such abstractions are not so readily achievable with other existing formalisms such as process algebras, labelled transition systems, and Petri nets, for example.

In order to model the example system from Section 3 using C^{2}KA, we first need to identify the support sets of the stimulus structure $\mathcal{S}$ and the CKA $\mathcal{K}$ that comprise the C^{2}KA. The support set of $\mathcal{S}$ is generated using the operations of $\mathcal{S}$ and the set of basic stimuli $\{start, load, loaded, prepare, done, unload, unloaded, setup, ready, process, processed, \mathfrak{d}, \mathfrak{n}\}$ where $\mathfrak{d}$ and $\mathfrak{n}$ represent the deactivation and neutral stimuli, respectively. The support set of $\mathcal{K}$ is generated using the operations of $\mathcal{K}$ and the

Table 1. Stimulus-response specification of the Control Agent C

○	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
IDLE	IDLE	IDLE	PREP	IDLE	IDLE	IDLE	IDLE	IDLE	IDLE	IDLE	IDLE
PREP	PREP	PREP	PREP	PREP	PREP	PREP	INIT	PREP	PREP	PREP	PREP
INIT	INIT	INIT	INIT	INIT	INIT	INIT	INIT	INIT	INIT	PROC	INIT
PROC	PROC	PROC	PROC	PROC	PROC	PROC	PROC	PROC	PROC	PROC	IDLE

λ	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
IDLE	*load*	n	*prepare*	n	n	n	n	n	n	n	n
PREP	n	n	n	n	n	n	*setup*	n	n	n	n
INIT	n	n	n	n	n	n	n	n	n	*done*	n
PROC	n	n	n	n	n	n	n	n	n	n	n

Table 2. Stimulus-response specification of the Storage Agent S

○	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
EMPTY	EMPTY	FULL	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY
FULL	FULL	FULL	FULL	FULL	FULL	EMPTY	FULL	FULL	FULL	FULL	FULL

λ	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
EMPTY	n	*loaded*	n	n	n	n	n	n	n	n	n
FULL	n	n	n	n	n	*unloaded*	n	n	n	n	n

Table 3. Stimulus-response specification of the Handling Agent H

○	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
WAIT	WAIT	WAIT	WAIT	MOVE	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT
MOVE	MOVE	MOVE	MOVE	MOVE	MOVE	MOVE	MOVE	MOVE	WAIT	MOVE	MOVE

λ	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
WAIT	n	n	n	*unload*	n	n	n	n	n	n	n
MOVE	n	n	n	n	n	n	n	n	*process*	n	n

Table 4. Stimulus-response specification of the Processing Agent P

○	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
STBY	STBY	STBY	STBY	STBY	STBY	STBY	STBY	SET	STBY	STBY	STBY
SET	SET	SET	SET	SET	SET	SET	SET	SET	SET	WORK	SET
WORK	WORK	WORK	WORK	WORK	STBY	WORK	WORK	WORK	WORK	WORK	WORK

λ	*start*	*load*	*loaded*	*prepare*	*done*	*unload*	*unloaded*	*setup*	*ready*	*process*	*processed*
STBY	n	n	n	n	n	n	n	*ready*	n	n	n
SET	n	n	n	n	n	n	n	n	n	*processed*	n
WORK	n	n	n	n	n	n	n	n	n	n	n

set of basic behaviors $\{$IDLE, PREP, INIT, PROC, EMPTY, FULL, WAIT, MOVE, STBY, SET, WORK, 0, 1$\}$ where 0 and 1 represent the inactive and idle behaviors, respectively.

Using the C^{2}KA described above, the behavior of each system agent is abstractly represented by:

$$\begin{aligned} \mathsf{C} &\mapsto \big\langle \text{IDLE} + \text{PREP} + \text{INIT} + \text{PROC} \big\rangle \\ \mathsf{S} &\mapsto \big\langle \text{EMPTY} + \text{FULL} \big\rangle \\ \mathsf{H} &\mapsto \big\langle \text{WAIT} + \text{MOVE} \big\rangle \\ \mathsf{P} &\mapsto \big\langle \text{STBY} + \text{SET} + \text{WORK} \big\rangle \end{aligned}$$

For example, this shows that the Control Agent C, at any given time, can exhibit any one of the four behaviors of idle, preparing, initializing, or processing, as denoted by the CKA term constructed using the non-deterministic choice operator $+$. Note that for the specification of more complex agent behaviors, the abstract behavior specification may include more complex CKA terms involving additional CKA operators to indicate sequential or parallel composition of behaviors from the CKA $\mathcal{K}$, for example. In this way, the use of the C^{2}KA framework for the modeling and specification of distributed systems provides the expressiveness to represent agent behaviors within a wide range of complexity.

We derive the *stimulus-response specification of agents* which specifies the next behavior mapping $\circ$ and next stimulus mapping λ for each system agent. The stimulus-response specification of the agents are derived from the description of the example in Section 3 and are shown in Tables 1–4. The row headers show the possible basic behaviors of the given agent, and the column headers show the basic stimuli to which the agent may be subjected. The grids show the next behavior or next stimulus that results when the stimulus in the column header is applied to the behavior in the row header.

4.2. Formulating implicit interactions

An interaction between two system agents can be understood as a potential for communication characterized by the existence of a communication path allowing for the transfer of data or control from one agent to another. Communication via stimuli from agent A to agent B (denoted by $\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B}$) is said to have taken place only when a stimulus generated

by A *influences* (i.e., causes an observable change, directly or indirectly) the behavior of B. Note that it is possible that more than one agent is influenced by the generation of the same stimulus by another agent in the system. The potential for communication via stimuli is formally defined within the C^{2}KA framework in [35] and [36] and is not repeated here.

An *implicit interaction* is an unforeseen, unexpected, or unintended communication path (influence) in a system. A system often has an intended sequence of communication to perform its function. Implicit interactions are those potential influences that are not explicitly stated as part of the intended system functionality. Let $\mathcal{P}_{\mathsf{intended}}$ be the set of intended interactions. This set of intended interactions can be derived from the system description and requirements explicitly provided by the system designer. For example, it may be represented by the system designer as a message passing diagram, similar to that shown in Figure 1, or alternatively, as a collaboration or sequence diagram.

Consider the manufacturing cell control system described in Section 3. By extracting the possible sequences of communication from the message passing diagram, the intended system interactions that comprise $\mathcal{P}_{\mathsf{intended}}$ can be derived. For example, the sequences depicted in Figure 1 can be unravelled to obtain Figure 2 which shows the control flow sequence among the system agents. Since the Processing Agent P and the Control Agent C respond to the *process* stimulus at the same time, the expansion of the concurrent interaction of the system agents is captured by the branching in the control flow sequence and in the set of intended interactions. In other words, the set of intended system interactions captures the possible execution traces representing the interleavings of the concurrent behaviors of the system agents. With respect to Figure 2 for the manufacturing cell control system example, the set of intended interactions can be characterized as follows:

$$\begin{aligned}\mathcal{P}_{\mathsf{intended}} \;=\; \{&\mathsf{C}\to\mathsf{S}\to\mathsf{C}\to\mathsf{H}\to\mathsf{S}\to\mathsf{C}\to\mathsf{P}\to\mathsf{H}\to\mathsf{P}\to\mathsf{C},\\ &\mathsf{C}\to\mathsf{S}\to\mathsf{C}\to\mathsf{H}\to\mathsf{S}\to\mathsf{C}\to\mathsf{P}\to\mathsf{H}\to\mathsf{C}\to\mathsf{P}\;\}\end{aligned}$$

We acknowledge that, in some cases, a complete specification or characterization of the intended system interactions may not be provided or may not be easily derived. In such cases, it may be possible to alternatively characterize the intended system interactions as a collection of properties of the modeled system. For example, we can have a property that expresses that agent A should not be able to communicate via stimuli with agent B (formally $\neg(\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B})$). However, such properties must be carefully specified in order to ensure that they are not overly restrictive or relaxed, and to ensure that the collection of properties does in fact completely characterize the intended system interactions. For this reason, we have elected to take a more systematic and rigorous approach for characterizing the intended system interactions, despite that for reasonably large systems this set may be quite large. We also note that it is possible to automate the process of identifying the set of intended system interactions that comprise $\mathcal{P}_{\mathsf{intended}}$ if we assume that we are given a specific representation (e.g., message passing diagram) of the expected (designed) system behavior. The articulation

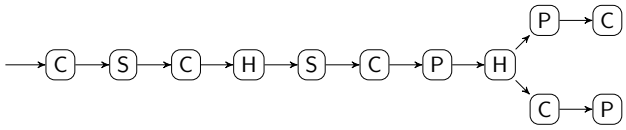

Figure 2. Expected control flow sequence depicting the intended system interactions captured by $\mathcal{P}_{\mathsf{intended}}$ for the manufacturing cell control system

of such a representation of the expected system behavior is typically part of any decent system engineering process. This kind of automation can help to alleviate the amount of manual effort required by system analysts in determining the set of intended system interactions.

Consider a system formed by a set $\mathcal{A}$ of agents with $\mathsf{A}, \mathsf{B} \in \mathcal{A}$ such that $\mathsf{A} \neq \mathsf{B}$. The existence of an implicit interaction via stimuli is formally defined in Definition 2.

Definition 2 (Existence of an Implicit Interaction via Stimuli). An *implicit interaction via stimuli* from agent A to agent B exists in a system if and only if:

$$\begin{aligned}\exists\big(p \;\mid\; p \implies (\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B}) \;:\;\\ \forall(q \;\mid\; q \in \mathcal{P}_{\mathsf{intended}} \;:\; \neg\mathrm{SubPath}(p,q))\big)\end{aligned}$$

where $\mathrm{SubPath}(p,q)$ is a predicate indicating that p is a subpath of q.

This is to say that if there exists a path p that indicates a potential for communication via stimuli from A to B and that is not a subpath of any of the intended interactions characterized by the set $\mathcal{P}_{\mathsf{intended}}$, then there exists at least one implicit interaction via stimuli in the system that is p.

4.3. Identifying implicit interactions

Given a system modeled using C^{2}KA and the stimulus-response specification of agents (see Tables 1-4), we use the C^{2}KA prototype tool described in [35] to run an analysis of the potential for communication via stimuli for the given system. The tool automatically provides a list of all potential communication paths via stimuli between each pair of agents in the system. Once we have identified the potential communication paths that can exist in the system, we verify whether each is an implicit interaction by applying Definition 2 with respect to the set of intended system interactions. For interactions p and q, this verification can be done with complexity $O(|p||q|)$ using dynamic programming approaches (e.g., [22]).

As an example, with respect to $\mathcal{P}_{\mathsf{intended}}$ given above, the tool output for the potential for communication from H to C yields the communication path $\mathsf{H} \rightarrow \mathsf{S} \rightarrow \mathsf{C}$ which is a subpath of one of the interactions in $\mathcal{P}_{\mathsf{intended}}$, meaning this interaction is intended and expected as part of the system behavior. However, when examining the potential for communication from P to S, there is a potential communication path $\mathsf{P} \rightarrow \mathsf{C} \rightarrow \mathsf{S}$ which does not exist as a subpath of any of the interactions in $\mathcal{P}_{\mathsf{intended}}$. This means that this path represents an implicit interaction and indicates that it

is possible for the Processing Agent P to indirectly influence the behavior of the Storage Agent S, despite having no interaction (direct or indirect) with respect to the given intended system behavior. Such an implicit interaction is possible due to the potential for out-of-sequence stimuli to be issued as a consequence of agents experiencing failures or being subject to malicious activity, for example.

We note that after having identified the presence of implicit interactions in a given system, each implicit interaction should be validated to ensure that it is indeed an unwanted behavior, rather than an undocumented expected behavior. Furthermore, due to the level of abstraction of the given system specification using C^2KA, we acknowledge that it is possible that the proposed approach identifies interactions which are considered to be implicit interactions, but that are very unlikely to manifest in the real world (e.g., they exist as model errors). Consequently, we identify the need for methods and techniques for analyzing the identified implicit interactions, such as measuring their severity, but also more sophisticated approaches that delve into the semantics of the identified implicit interactions in order to validate their existence, and to more accurately assess the threat that they pose to the overall safety and security of the given system. Such validation can be done by incorporating system designers in the loop, for example, and can involve simulations of the C^2KA model of the system through the development of scenarios where a particular agent, from which an identified implicit interaction originates, sends a stimulus, or sequence of stimuli, other than that which is it is expected to send, and by examining the resulting system behaviors which can be computed (automatically) using the axiomatic system of C^2KA.

4.4. Experimental results

After analyzing each of the potential communication paths between each pair of agents in the manufacturing cell control system example described in Section 3, we found that 11 of the 30 total possible communication paths between each pair of agents are not found as a subpath of any of the interactions in $\mathcal{P}_{\mathsf{intended}}$ and are therefore implicit interactions. A summary of our experimental results which are automatically generated with the use of a prototype tool is given in Table 5. It is important to note that these implicit interactions are not easily found without the use of the formal analysis of the system based on its C^2KA specification. The illustrative example shows that, even for a very small system, there is hidden complexity and coupling among agents that can lead to the potential for unexpected system behaviors.

Implicit interactions are undesirable since they offer a means for system agents to interact in unintended ways. Their existence indicates that there is an aspect of the system design (whether intentional or accidental, innocuous or malicious) allowing for this kind of interaction to be present. These interactions constitute linkages within a system of which designers are generally unaware, and that therefore constitute a security vulnerability. These vulnerabilities can be exploited if a user can gain access to the component from which the implicit interaction originates. For instance, a malicious agent can force stimuli to be sent out-of-sequence in order to cause the system to behave in a particular way that may have severe consequences in terms of safety and security. By demonstrating the existence of implicit interactions and the resulting unintended influence of agents, we expect that for larger, more complex, and more safety-critical systems exhibiting the same characteristics, the methods presented here will be able to demonstrate the specific manner in which these influences could be triggers for very significant destabilizing events.

Table 5. Summary of experimental results for identifying implicit interactions in the manufacturing cell control system

Interaction	# Implicit Interactions	# Total Possible Paths
$\mathsf{C} \rightarrow_{\mathcal{S}}^{+} \mathsf{H}$	0	2
$\mathsf{C} \rightarrow_{\mathcal{S}}^{+} \mathsf{P}$	1	2
$\mathsf{C} \rightarrow_{\mathcal{S}}^{+} \mathsf{S}$	1	3
$\mathsf{H} \rightarrow_{\mathcal{S}}^{+} \mathsf{C}$	0	3
$\mathsf{H} \rightarrow_{\mathcal{S}}^{+} \mathsf{P}$	0	3
$\mathsf{H} \rightarrow_{\mathcal{S}}^{+} \mathsf{S}$	2	3
$\mathsf{P} \rightarrow_{\mathcal{S}}^{+} \mathsf{C}$	1	3
$\mathsf{P} \rightarrow_{\mathcal{S}}^{+} \mathsf{H}$	1	2
$\mathsf{P} \rightarrow_{\mathcal{S}}^{+} \mathsf{S}$	4	4
$\mathsf{S} \rightarrow_{\mathcal{S}}^{+} \mathsf{C}$	0	1
$\mathsf{S} \rightarrow_{\mathcal{S}}^{+} \mathsf{H}$	0	2
$\mathsf{S} \rightarrow_{\mathcal{S}}^{+} \mathsf{P}$	1	2
TOTAL	11	30

5. Discussion

There are increasing concerns related to the cybersecurity of modern computer systems and networks, and assuring the safety, security, and reliability of cyber-physical systems remains among the top priorities for governments and providers of communications, financial, electric, and other services (e.g., [6], [65]). Many of the distributed cyber-physical systems operating within critical infrastructures and other safety-critical domains are open systems, meaning that they participate in intensive communication and exchange with their environment, which often includes other systems. These kinds of systems need input in terms of energy, resources, information, etc., and as a result, the interactions between a system and its environment need to be carefully taken into account when modeling such systems [41].

By contrast to existing work (see Section 2) which focusses a lot of attention on analyzing systems at later stages in the development, the proposed approach for identifying and analyzing implicit interactions provides a step towards

uncovering potential cybersecurity vulnerabilities resulting from the existence of implicit interactions in distributed cyber-physical systems. It is a rigorous formal methods-based approach using an algebraic modeling framework capable of identifying the implicit interactions that may be present in a system with respect to a given specification early in the design stage of system development. This gives us a way in which we can analyze system designs in order to identify cybersecurity vulnerabilities and weaknesses so that these design flaws may be addressed before they are able to manifest in a system after it has been implemented and deployed. It also aids in understanding the risks being taken if such vulnerabilities are not dealt with. Furthermore, the proposed approach provides a different perspective of a system and the problem of implicit interactions by offering a different level of abstraction that allows us to consider a variety of systems and components/agents. This gives an alternative way in which we can view the interactions of systems and agents than what has been traditionally provided by formalisms such as process algebras, and approaches such as non-interference and information flow analysis. More broadly, the proposed approach adds to and complements existing work that helps us to understand and assure the safety and security of complex systems.

5.1. Scalability of the proposed approach

Another important issue which presents both challenges and opportunities with respect to the proposed approach is its scalability and complexity. Factors such as the number of agents, the number of stimuli, the number of basic behaviors, as well as the complexity of each agent behavior, all contribute to the overall complexity of the system. Without a doubt, real world systems are highly complex and scale in a very unfriendly way, and we need to be able to handle such issues.

Certainly, direct applicability of the proposed approach to a system consisting of millions of agents would be impractical. However, large systems can still be modeled successfully with the appropriate treatment of agents and their behaviors. For example, by taking a more coarse-grained view of the system and aggregating the behaviors of a group of agents into a single agent, which is straightforward due to the abstract nature of the C^2KA framework, we can reduce the overall complexity of the system under consideration. This idea is reflected in the illustrative example described throughout this paper as the Processing Agent P is itself comprised of a numerous different agents (e.g., within a cyber-physical manufacturing system) responsible for performing the specific processing tasks of the manufacturing process. However, for the purpose of analyzing interactions, it was appropriate to treat the Processing Agent as a single entity. Generally speaking, a significant number of the systems that can be analyzed using the proposed approach can be decomposed and structured hierarchically in order to reduce the number of distinct agents. Moreover, the proposed algorithms and problem data sets for identifying implicit interactions can be parallelized in a straightforward manner. In addition, the use of sophisticated data representations, such as binary decision diagrams (BDDs) [32], can address a number of the issues surrounding the amount of computational power and space required to analyze larger systems. Therefore, we argue that the proposed approach can scale to handle larger and more complex systems, but due to resource limitations, we do not provide a full scalability study in this paper. However, as a priority for our future work, we identify the need for setting up a simulation environment and workbench to study the practical effectiveness and scalability of our approach in order to obtain estimates on the bounds of the size and complexity of systems for which our approach can feasibly identify implicit component interactions.

6. Concluding remarks and future work

We have presented an approach for identifying implicit interactions in distributed cyber-physical systems in an effort to address an important open question (e.g., [6]) related to the application of formal methods to protecting critical systems from cyber-threats. The approach is based on the specification and analysis of a system using the C^2KA framework. It is meant to be used as a tool for analyzing the designs of distributed cyber-physical systems and to guide the effort of designing-in security through the development of approaches for mitigating the existence and threat of implicit interactions in system designs. Our results show that a substantial fraction of the potential interactions represented as communication paths can exist as implicit interactions that may be unintended or unforeseen by the system designer. Consequently, we have demonstrated that the proposed approach can help to specify, design, and implement systems with improved cybersecurity assurance.

In our current and future work, we are developing a framework to characterize the severity of implicit interactions, as that information can be invaluable in identifying where to focus efforts aimed at improving system stability and security. In addition, we intend to provide simulations of cyber-attacks mounted upon implicit interactions to study the effects in their systems. Furthermore, we aim to further examine the scalability of the proposed approach using case studies of more complex models inspired by critical infrastructure systems. Subsequently, we plan to investigate solutions, like those proposed in [33], for mitigating implicit interactions in distributed cyber-physical systems.

Acknowledgments

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number, 2015-ST-061-CIRC01.

Disclaimer: The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] P. A. Abdulla, F. Haziza, and L. Holík, "All for the price of few (parameterized verification through view abstraction)," in *Proceedings fo the 14th International Conference on Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, R. Giacobazzi, J. Berdine, and I. Mastroeni, Eds. Springer Berlin Heidelberg, 2013, vol. 7737, pp. 476–495.

[2] K. Alghathbar, C. Farkas, and D. Wijesekera, "Securing UML information flow using FlowUML," *Journal of Research and Practice in Information Technology*, vol. 38, no. 1, pp. 111–120, February 2006.

[3] R. Alur and T. A. Henzinger, "Reactive modules," *Formal Methods in System Design*, vol. 15, no. 1, pp. 7–48, 1999.

[4] G. R. Andrews and R. P. Reitman, "An axiomatic approach to information flow in programs," *ACM Transactions on Programming Languages and Systems*, vol. 2, no. 1, pp. 56–76, January 1980.

[5] H. Baker and C. Hewitt, "Laws for communicating parallel processes," Massachusetts Institute of Technology, AI Working Paper 134A, May 1977.

[6] C. Bennett, "Feds lack method to grade critical infrastructure cybersecurity," Available: http://thehill.com/policy/cybersecurity/260963-feds-lack-method-to-grade-critical-infrastructure-cybersecurity, November 2015.

[7] J. Bergstra and J. Klop, "Process algebra for synchronous communication," *Information and Control*, vol. 60, no. 1-3, pp. 109–137, 1984.

[8] S. Chong and R. Van Der Meyden, "Using architecture to reason about information security," *ACM Transactions on Information and System Security*, vol. 18, no. 2, pp. 8:1–8:30, December 2015. [Online]. Available: http://doi.acm.org/10.1145/2829949

[9] E. Clarke, M. Talupur, and H. Veith, "Environment abstraction for parameterized verification," in *Proceedings of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, E. A. Emerson and K. S. Namjoshi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 3855, pp. 126–141.

[10] W. D. Clinger, "Foundations of actor semantics," Massachusetts Institute of Technology, Tech. Rep. 633, June 1981.

[11] V. Debruyne, F. Simonot-Lion, and Y. Trinquet, *Architecture Description Languages*. Boston, MA: Springer, 2005, ch. EAST-ADL — An Architecture Description Language, pp. 181–195.

[12] D. E. Denning, "A lattice model of secure information flow," *Communications of the ACM*, vol. 19, no. 5, pp. 236–243, May 1976.

[13] D. D. Dudenhoeffer, M. R. Permann, and M. Manic, "CIMS: A framework for infrastructure interdependency modeling and analysis," in *Proceedings of the 2006 Winter Simulation Conference*, L. F. Perrone, F. P. Wiel, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, Eds., 2006, pp. 478–485.

[14] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction," Carnegie Mellon University Software Engineering Institute, Tech. Rep. CMU/SEI-2006-TN-011, February 2006.

[15] R. Focardi and R. Gorrieri, "A classification of security properties for process algebras," *Journal of Computer Security*, vol. 3, no. 1, pp. 5–33, November 1994.

[16] R. Focardi, R. Gorrieri, and F. Martinelli, "Real-time information flow analysis," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 1, pp. 20–35, 2003.

[17] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*. Elsevier, 2011.

[18] S. M. German and A. P. Sistla, "Reasoning about systems with many processes," *Journal of the ACM*, vol. 39, no. 3, pp. 675–735, July 1992.

[19] J. Goguen and J. Meseguer, "Security policies and security models," in *Proceedings of the 1982 Symposium on Security and Privacy*, New York, NY, U.S.A., 1982, pp. 11–20.

[20] I. Greif, "Semantics of communicating parallel processes," Ph.D. dissertation, Massachusetts Institute of Technology, August 1975.

[21] J. Gu, J.-l. Zhou, S.-s. Yu, J. Zhang, and Z.-c. Duan, "Study on the architecture of control system for manufacturing cells," *Journal of Shanghai University (English Edition)*, vol. 5, no. 2, pp. 130–135, June 2001.

[22] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997. [Online]. Available: https://books.google.com/books?id=Ofw5w1yuD8kC

[23] R. Hähnle, J. Pan, P. Rümmer, and D. Walter, "Integration of a security type system into a program logic," *Theoretical Computer Science*, vol. 402, no. 2–3, pp. 172–189, 2008.

[24] J. T. Haigh, R. A. Kemmerer, J. McHugh, and W. D. Young, "An experience using two covert channel analysis techniques on a real system design," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 157–168, February 1987.

[25] L. Hélouët, M. Zeitoun, and A. Degorre, "Scenarios and covert channels: Another game..." *Electronic Notes in Theoretical Computer Science*, vol. 119, pp. 93–116, 2005.

[26] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular ACTOR formalism for artificial intelligence," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, 1973, pp. 235–245.

[27] C. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, August 1978.

[28] C. Hoare, B. Möller, G. Struth, and I. Wehrman, "Concurrent Kleene algebra and its foundations," *Journal of Logic and Algebraic Programming*, vol. 80, no. 6, pp. 266–296, 2011.

[29] C. Hoare and J. Wickerson, "Unifying models of data flow," in *Proceedings of the 2010 Marktoberdorf Summer School on Software and Systems Safety*, M. Broy, C. Leuxner, and C. Hoare, Eds. IOS Press, August 2011, pp. 211–230.

[30] K. Hristova, T. Rothamel, Y. A. Liu, and S. D. Stoller, "Efficient type inference for secure information flow," in *Proceedings of the 2006 Workshop on Programming Languages and Analysis for Security*, ser. PLAS '06, New York, NY, U.S.A., October 2006, pp. 85–94.

[31] M. Iturbe, I. Garitano, U. Zurutuza, and R. Uribeetxeberri, "Visualizing network flows and related anomalies in industrial networks using chord diagrams and whitelisting," in *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, vol. 2, 2016, pp. 99–106.

[32] G. Janssen, "A consumer report on BDD packages," in *Proceedings of the 16th Symposium on Integrated Circuits and Systems Design*, ser. SBCCI 2003. Washington, DC, U.S.A.: IEEE Computer Society, September 2003, pp. 217–222.

[33] J. Jaskolka and R. Khedri, "Mitigating covert channels based on analysis of the potential for communication," *Theoretical Computer Science*, vol. 643, pp. 1–37, August 2016.

[34] J. Jaskolka, R. Khedri, and Q. Zhang, "Endowing concurrent Kleene algebra with communication actions," in *Proceedings of the 14th International Conference on Relational and Algebraic Methods in Computer Science*, ser. Lecture Notes in Computer Science, P. Höfner, P. Jipsen, W. Kahl, and M. Müller, Eds. Springer International Publishing Switzerland, 2014, vol. 8428, pp. 19–36.

[35] J. Jaskolka, "On the modelling, analysis, and mitigation of distributed covert channels," Ph.D. dissertation, McMaster University, Hamilton, ON, Canada, March 2015, available: http://hdl.handle.net/11375/16872.

[36] J. Jaskolka and R. Khedri, "A formulation of the potential for communication condition using C^2KA," in *Proceedings of the 5th International Symposium on Games, Automata, Logics and Formal Verification*, ser. Electronic Proceedings in Theoretical Computer Science, A. Peron and C. Piazza, Eds. Verona, Italy: Open Publishing Association, September 2014, vol. 161, pp. 161–174.

[37] A. A. E. Kalam, Y. Deswarte, A. Baïna, and M. Kaâniche, "Polyorbac: A security framework for critical infrastructures," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 4, pp. 154–169, 2009.

[38] R. M. Keller, "Formal verification of parallel programs," *Communications of the ACM*, vol. 19, no. 7, pp. 371–384, July 1976.

[39] N. Kobayashi, "Type-based information flow analysis for the π-calculus," *Acta Informatica*, vol. 42, no. 4, pp. 291–347, 2005.

[40] D. Kozen, "On action algebras," in *Logic and Information Flow*. MIT Press, 1993, pp. 78–88.

[41] W. Kröger and E. Zio, *Vulnerable Systems*. London: Springer London, 2011, ch. 3: Challenges to Methods for the Vulnerability Analysis of Critical Infrastructures, pp. 33–39.

[42] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 2, pp. 125–143, March 1977.

[43] A. Letichevsky and D. Gilbert, "A general theory of action languages," *Cybernetics and Systems Analysis*, vol. 34, pp. 12–30, 1998.

[44] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2011.

[45] T. G. Lewis, *Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation*. John Wiley & Sons, Inc., 2006.

[46] G. Lowe, "Quantifying information flow," in *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, ser. CSFW-15. Los Alamitos, CA, U.S.A.: IEEE Computer Society, 2002, pp. 18–31.

[47] J. McDermid and Q. Shi, "A formal model of security dependency for analysis and testing of secure systems," in *Proceedings of the Computer Security Foundations Workshop IV*, Los Alamitos, CA, U.S.A., 1991, pp. 188–200.

[48] R. Milner, *A Calculus of Communicating Systems*, ser. Lecture Notes in Computer Science. Springer-Verlag, 1980, vol. 92.

[49] ——, *Communication and Concurrency*, ser. Prentice-Hall International Series in Computer Science. Prentice Hall, 1989.

[50] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes part I," *Information and Computation*, vol. 100, no. 1, pp. 1–40, September 1992.

[51] K. S. Namjoshi and R. J. Trefler, "Analysis of dynamic process networks," in *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, C. Baier and C. Tinelli, Eds. Springer Berlin Heidelberg, 2015, vol. 9035, pp. 164–178.

[52] ——, "Parameterized compositional model checking," in *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, M. Chechik and J.-F. Raskin, Eds. Springer Berlin Heidelberg, 2016, vol. 9636, pp. 589–606.

[53] M. Ouyang, "Review on modeling and simulation of interdependent critical infrastructure systems," *Reliability Engineering & System Safety*, vol. 121, pp. 43–60, 2014.

[54] C. A. Petri, "Kommunikation mit automaten," Ph.D. dissertation, Institut für instrumentelle Mathematik, Bonn, Germany, 1962, English translation available as: *Communication with Automata*, Technical Report RADC-TR-65-377, volume 1, supplement 1, Applied Data Research, Princeton, NJ, U.S.A., 1966.

[55] V. Pratt, "Action logic and pure induction," in *Logics in AI*, ser. Lecture Notes in Computer Science, J. Eijck, Ed. Springer Berlin/Heidelberg, 1991, vol. 478, pp. 97–120.

[56] S. M. Rinaldi, "Modeling and simulating critical infrastructures and their interdependencies," in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004*, January 2004, pp. 1–8.

[57] P. Y. A. Ryan and S. A. Schneider, "Process algebra and non-interference," in *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, 1999, pp. 214–227.

[58] P. Ryan, J. McLean, J. Millen, and V. Gligor, "Non-interference: Who needs it?" in *Proceedings of the 14th IEEE Workshop on Computer Security Foundation*. Washington, DC, U.S.A.: IEEE Computer Society, 2001, pp. 237–238.

[59] K. E. Sabri, R. Khedri, and J. Jaskolka, "Verification of information flow in agent-based systems," in *Proceedings of the 4th International MCETECH Conference on e-Technologies*, ser. Lecture Notes in Business Information Processing, G. Babin, P. Kropf, and M. Weiss, Eds., vol. 26. Springer Berlin/Heidelberg, May 2009, pp. 252–266.

[60] K. Schweiker, S. Varadarajan, D. Spivak, P. Schultz, R. Wisnesky, and M. Pérez, "Operadic analysis of distributed systems," National Aeronautics and Space Administration, Tech. Rep. NASA/CR–2015–xxxxxx, September 2015.

[61] A. Shaffer, M. Auguston, C. Irvine, and T. Levin, "Toward a security domain model for static analysis and verification of information systems," in *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling*, ser. DSM '07, Montreal, QC, Canada, October 2007, pp. 160–171.

[62] J. Shen and S. Qing, "A dynamic information flow model of secure systems," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '07. Singapore: ACM, 2007, pp. 341–343.

[63] S. F. Siegel and G. Gopalakrishnan, "Formal analysis of message passing," in *Proceedings of the 12th International Conference on Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, R. Jhala and D. Schmidt, Eds. Springer Berlin Heidelberg, 2011, vol. 6538, pp. 2–18.

[64] United States Department of Defense, "Procedures for performing a failure mode effect and criticality analysis," Department of Defense, Tech. Rep. MIL-STD-1629A, November 1980.

[65] U.S.A. Department of Homeland Security, "A roadmap for cybersecurity research," Department of Homeland Security Science and Technology Directorate, Washington, DC, U.S.A., November 2009.

[66] R. Van Der Meyden, "What, indeed, is intransitive noninterference?" in *Proceedings of the 12th European Symposium On Research In Computer Security*, J. Biskup and J. López, Eds. Springer Berlin Heidelberg, 2007, pp. 235–250.

[67] V. Varadharajan, "Petri net based modelling of information flow security requirements," in *In Proceedings of the Computer Security Foundations Workshop III*, June 1990, pp. 51–61.

[68] D. Volpano and G. Smith, "Eliminating covert flows with minimum typings," in *Proceedings of the 10th Computer Security Foundations Workshop*, Los Alamitos, CA, U.S.A., 1997, pp. 156–168.

[69] D. Volpano, G. Smith, and C. Irvine, "A sound type system for secure flow analysis," *Journal of Computer Security*, vol. 4, no. 2-3, pp. 167–187, 1996.

[70] G. Winskel, "Event structures," in *Petri Nets: Applications and Relationships to Other Models of Concurrency*, ser. Lecture Notes in Computer Science, W. Brauer, W. Reisig, and G. Rozenberg, Eds. Springer Berlin/Heidelberg, 1987, vol. 255, pp. 325–392.