An algebra for interaction of cyber-physical components

Lion, B.

Chapter 1

Introduction

1.1 Context

Today's technological and theoretical progress bring new challenges in the field of theoretical computer science. The emergence of digital systems that reliably measure and actuate physics reveals new kinds of interconnected systems that we call *cyber-physical systems.* A cyber-physical system typically refers to a system in which digital processes (e.g., controllers) interact (e.g., via sensing or actuating) with and through physical medium (e.g., space, time). The understanding of cyber-physical systems is branches into several lines of research [57]. We should mention first the field of cybernetics, whose appearance in the literature dates back to Wiener [89]. Cybernetics, and more largely control theory, aim at studying the feedback mechanisms taking place between a governor (kuberneties in Greek) and a physical system. Given a suitable model of how the physics operates, the governor can *steer* the physical system towards a desired objective and control its behavior [79, 25]. Also, and on top of cybernetics, the interaction among cyber-physical systems is a concern in the field of emergent behavior and swarm robotics [39]. The objective is to find suitable coordination patterns that enable a set of interacting cyber-physical systems to reach a collective objective. Still, challenges remain in the design and analysis for complex cyber-physical systems, and the concurrency that produces in the interaction among their cyber and physical parts. The specific challenge that we undertake in this thesis is that of a design framework that is *compositional* and *concurrent.* We describe in more details the essence and implications of two properties that we deem essential in formal models for cyber-physical systems.

Compositionality is, intuitively, the property that allows forming a *complex* system by assembling *simpler* systems together. Compositionality is used as a principle [83, 44], for instance, in order to assign meanings of natural language sentences given the meaning of its part. This principle has permeated many aspects of computer science, such as in program semantics, e.g., assigning meanings to programs, and in system design, e.g., defining a framework to construct systems. Given a set of small blocks, and a rule to assemble such blocks, one can quickly get a system whose behavior surpasses in complexity each of its parts. The emergence of new kinds of machines (e.g., interconnected networks, cyber-physical systems) requires a new stand on the question of compositional specification for such systems (see Chapter 2).

Concurrency is the field in theoretical computer science that studies the behavior of a set of communicating processes. The aim is to understand the behavior emerging from a set of interacting machines as yet another machine, similarly to how the behavior of a machine would be understood from its parts. One of the challenges, for instance, comes from the fact that, in a network, events may occur in parallel, independently, or *at the same time*. Therefore, the mode of interaction among machines is not necessarily derivable from the behavior specification of each machine. Several models of how machines interact through communication have been studied, from the original neural network model [54], or the network of communicating machines [70]. While still many paths explore ways to design and analyze concurrent systems, we observe yet another type of concurrent systems of interest: in cyber-physical systems, the physical medium in which machines evolve plays an active role in the interaction among the cyber parts.

We give an illustrative example to justify that compositionality and concurrency are two important features to include in a model for cyber-physical systems. For instance, consider a group of robots, each running a program that takes decision based on the sequence of sensor readings. The sensors that equip a robot return the current position of the robot and the position of any adjacent obstacle. The interaction occurring between robots in the group cannot be derived solely from the specification of individual robots. If the field on which the robots roam changes its property, the same group of robots might sense different values, and therefore take different actions. Also, the time at which a robot acts and senses will affect the decision of each controller and will change the resulting collective behavior.

Other instances of applications emerge from the challenges to architect a cyber-physical system, as in the trends of Industry 4.0 [82], digital twins [19], or the Internet of Things [10]. In Industry 4.0, the manufacturing process is equipped with sensors, that observe physical variables of interest along the supply chain, and actuators that perform physical tasks. Such architecture aims at improving the automation of repetitive physical tasks, the allocation of resources, but also the detection and signaling of malfunctioning devices. The digital twin trend aims at creating models of physical phenomenon that gather information, update, and monitor in live physical objects to achieve some objectives. In such cases, having a formal design framework to model interaction between the physics and its discrete model is of key importance to minimize faults and increase accuracy. The Internet of Things captures the idea of connecting sensors and actuators over the internet. Each device therefore becomes a node that has cyber-physical capabilities that can be addressed remotely. As a result, the protocol that rules the interaction over the nodes in such a network is central to prove properties of, for instance, security or resilience.

Generally, a design framework gives some means to specify *what* behavior is desired and hides internal details that explain *how* a (network of) machines would construct such behavior. As pictured in Figure 1.1 the operational part of a system, which is represented by machines M_1 and M_2 interacting under a protocol Σ, is separated from the description of its behavior, which is given by the transformation $[\![\cdot]\!]$ as components C_1 and C_2. Ideally, operations on machine behaviors, such as the algebraic operations $\times_\Sigma$, should reflect practical interactions between machines. If such is the case, the behavior of the system consisting of machines M_1 and M_2 under the protocol Σ, is formally captured by the product of components C_1 and C_2 under the operation $\times_\Sigma$. The parameter Σ in the algebra is a new perspective that this thesis puts forward. This approach differs from, for instance, existing compositional models (e.g., hybrid IO automata [79], or hybrid programs [75]) that fix the parameter Σ from within the model, and expect each machine and component to already include the primitives to follow the protocol described by Σ.

One benefit of having such component algebra is that it allows for reasoning about updates. For instance, an update that preserves the behavior of the composition is a substitution of one of the components, such that the resulting collective behavior is unchanged. As a consequence, the class of all M_2 that preserves the same collective behavior C_3, under protocol Σ, contains possible machine replacements.

We highlight some fundamental differences in the interaction occurring between purely cyber components (e.g., discrete programs interacting with other discrete programs), and cyber-physical components (e.g., discrete programs interacting with physics). We leave the definition of a component and an interaction signature abstract, and refer to Chapter 2 for a more precise description.

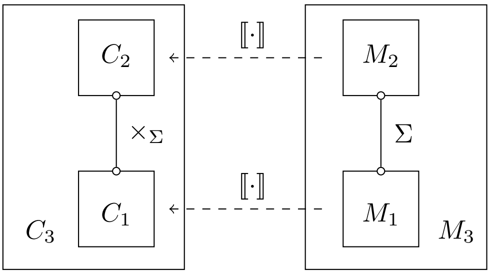

Figure 1.1: Two machines M_1 and M_2, whose respective behavior is captured by C_1 and C_2, interact through a protocol Σ. The product of the two components under the operation $\times_\Sigma$ reflects the behavior of the machines interacting with the protocol Σ.

Cyber-cyber interaction The models that capture interactions between two cyber components follows, in general, several assumptions:

- *Reproducibility.* The time value at which two machines M_1 and M_2 initially start does not change the resulting behavior. This assumption also means that the same protocol between the two machines is independent of the initial time, and therefore reproducible at a later time.

- *Closed system.* Given a fixed set of machines and a fixed protocol among the machines, one can, in theory, reason about the whole system statically. In a closed system, the composition of each machine's description under the interaction protocol provides a full description of the system state space. Static analysis may therefore detect beforehand, for instance, some race conditions.

- *No event missed.* It is possible to coordinate each machine so that no interacting event is missed (i.e., no message exchanged is missed). For instance, the hardware connection between each machine assumes a frequency of communication that eliminates the possibility to miss an event on either end.

In Chapter 3, we explore the property of cyber-cyber interactions by studying the class of components for which only the ordering of events matters, and not the precise value of the observation's time stamp.

Cyber-physical interaction The assumptions underlying models for cyber-physical interactions are different in nature than assumptions for cyber-cyber interactions:

- *Variance under time shift.* Some physical systems have time dependent and chaotic responses while interacting with cyber systems. In such cases, the time at which the system is initialized changes the resulting behavior of the composite cyber-physical system. For instance, the program that controls the decision of an autonomous car will receive different responses from the physics if executed at different times (change of traffic, change of weather, change of landscape, etc.).

- *Open system.* Some physical systems are so complex that their analytical characterization is infeasible. As a consequence, some reasoning can happen only dynamically as reaction, and the resulting specification must adapt to unexpected alternatives.

- *Approximation and missed events.* The sensing of physical quantities is inherently lossy as such quantities are continuously changing (i.e., function of time). An observation captures samples of those quantities at a time instant. As a result, observable properties on sensor readings are not sufficient to infer that the underlying physical quantity satisfies as well such property. Mechanisms for detection of deviations and diagnosis are therefore necessary to catch errors at runtime due to approximative measurements.

- *Non uniform description.* Dynamics in physics is usually described using differential equations, and leads to characterizing the evolution of physical quantities over real numbers. Alternatively, digital systems make use of discrete measures, and clocked processors that time their sensing and actuations. Thus, modelling concurrency between physical processes and cyber machines gives rise to the problem of uniformly describing their interactions.

1.2 Structure

This thesis lays a foundation to tackle the challenges stated above. We record in the following list the main points of each chapter. We give, for each chapter, a short summary on how our results compare with the state of the art.

In Chapter 2, we present an algebra of components that can model the four points presented above, making it suitable for modelling interaction in cyber-physical systems. More precisely, components are primitives in this algebra, and capture time-sensitive behavior of a part of a system, which includes both cyber and physical aspects. A component, in isolation, denotes all possible sequences of observations over time that a machine or physical process can exhibit. In composition with other components, only some of such sequences will remain possible. The same component, within different *contexts*, results in different behaviors as in an open system.

The relation between event occurrences in the behavior of two components is captured by algebraic operators. Each operation of the algebra is parametrized by an interaction signature, that specifies how two components interact. Such interaction may for instance allow or disallow events to occur independently (or simultaneously) between two components. The same two components under different interactions would expose different resulting behavior. The algebra therefore allows in some cases for decomposition, using a suitable division operation.

In Section 2.3, we define an operation of linearization that transforms a transactional component, i.e., observing multiple events at the same time, to a linear component, i.e., observing a single event at a time. We give conditions that a valid linearization must satisfy and present two instances of valid linearization: one lock-step procedure that linearizes all observations of a transactional component, and one multi-round procedure that allows for some interleaving. In both cases, the linearization can be performed in parts, i.e., linearization distributes over the product on transactional components. The material in this chapter is based on two journal publications and a paper to be submitted:

- Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: *A semantic model for interacting cyber-physical systems.* J. Log. Algebraic Methods Program. 129: 100807 (2022)

- Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: *A formal framework for distributed cyber-physical systems.* J. Log. Algebraic Methods Program. 128: 100795 (2022)

- Kasper Dokter, Benjamin Lion, Hans-Dieter A. Hiep: *Compositional Linearizations of Transactional Behaviors.* To be submitted (2022)

Other models for cyber-physical systems exist, such as hybrid systems (e.g., Hybrid Programs [75], Hybrid automata [40, 65, 79]), and our semantic model differs and complements existing work in, at least two main points. First, we model interaction externally, as constraints that apply on the behavior of each component. Interaction is not limited to input/outputs as in most hybrid descriptions, and the difference between cyber and physical aspects is abstracted in the general concept of a component. The generality of the semantic model enables to give a *specification* of a component (such as M_1 and M_2 in Figure 1.1) as a hybrid program, or as an I/O hybrid automata, and define suitable composition operators in the algebra to compositionally define cyber-physical systems. Second, we choose to model the interaction occuring between components in a discrete way, as sequences of observations: we choose to model the continuity of physical systems within their description as a set of discrete sequences of observations. This description closely represents runtime observable behaviors of cyber-physical systems, and highlights new challenges such as proving that a cyber-physical system is safe when considering safety of runtime observables only.

In Chapter 3, we study a class of components for which the precise time-stamp value of the observation does not matter, under a product that models synchronous interaction. We express, in Section 3.1, the semantics of the Reo coordination language as an algebra of order sensitive components. As an example, we express some well known connectors as a product of port components, under a suitable interaction signature. In Section 3.3, we study temporal properties of Reo connectors. We give a procedure to generate a specification in the Promela language from a logical specification of Reo connectors. We verify temporal properties of Reo connector by using its Promela translation and the Spin model checker. Through this work, we identify some key constructs to simplify the specification of a temporal property using Linear Temporal Logic (LTL) given the port and memory primitives in Reo. The material of the third chapter is based on a workshop paper and an implementation, respectively:

- Benjamin Lion, Samir Chouali, Farhad Arbab: *Compiling Protocols to Promela and Verifying their LTL Properties.* MoDELS (Workshops) 2018: 31-39
- Benjamin Lion, *Treo to Promela compiler*, 10.5281/zenodo.7393621 (2018)

This work expands existing work on the Reo coordination language. More specially, we should mention the existing works on Reo semantics [47], on Reo compilers [46] and frameworks to verify temporal properties of Reo circuits [42]. Our work differs from existing work in two main points. First, we give an algebraic semantics for Reo whose primitive components are not channels, but ports. Basic Reo channels (such as a sync or fifo channel) can be described as an algebraic product of their ports, parametrized by the proper interaction signature. Moreover, the algebraic properties of the interaction signature provides new ways to reason about equivalent Reo expressions. Second, we give a logical specification of Reo channels and internal composition that minimizes the size of the resulting composition. The internal representation of a Reo circuit is then translated either to a model checker for temporal verification, or to an imperative language for execution. The tool on which this section is based has shown state of the art results.

In Chapter 4, we provide an operational and executable specification of components. We first introduce in Section 4.1 an intermediate state-based representation of a component behavior as a labeled transition system called a TES transition system. We define a wide class of products of two such transition systems, each parametrized by a composability relation on their observations. We show that the semantics of TES transition systems as components is compositional with respect to their parametrized product. Based on the intermediate TES transition system, we give in Section 4.2 an executable finite specification of components as agents specified in rewriting logic. The behavior resulting from a concurrent run of a set of agents depends on the composability relation that governs the interaction among the agent. We show that, for some composability relations, the behavior of the concurrent execution of agents coincides with the behavior of the product of the components representing those agents.

The three sections are based on three publications:

- Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: *Runtime Composition Of Systems of Interacting Cyber-Physical Components.* In proceeding WADT (2022)

- Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: *A Rewriting Framework for Cyber-Physical Systems.* In proceeding Isola (2022)

- Tobias Kappé, Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: *Soft component automata: Composition, compilation, logic, and verification.* Sci. Comput. Program. 183 (2019)

Our work relates to existing work on state space description of cyber-physical systems, such as hybrid automata [40, 65, 79], and rewriting framework of preference aware agents [85]. We introduce three differences. First, our state space specification of components is compositional with respect to a large set of composition operators on components. The structure of the interaction is therefore preserved when giving a specification for each component. Second, we make explicit some criteria for forming the product step-by-step at runtime, while avoiding deadlock and ensuring fairness. Third, we include preferences to our specification, which are necessary when considering large specification for open systems. We also provide mechanisms to propagate preferences through composition.

In Chapter 5, we give a concrete implementation of agents described in Chapter 4 in order to analyse collective behaviors resulting from their interaction. We implement in Section 5.1 the executable rewriting framework in Maude, and give a series of detailed applications to demonstrate the usefulness of the modeling framework and the scope of analysis that are possible. In Section 5.2, we implement a concurrent version of Reo and show that the framework is suitable for concurrent execution of Reo primitives. We analyze in Section 5.3 the protocol governing the interactions among a controller, a valve, and some reservoirs, by showing the safety of a controller's strategy. We verify in Section 5.4 some liveness, safety, and sorting properties for energy aware robots roaming on the same field. The material of this section is based on the following implementation:

- Benjamin Lion, *Cyber-physical agent framework in Maude*, Zenodo, 10.5281/zenodo.6592275 (2022)

The implementation is inspired from [53], which is a framework for detecting deviations in the concurrent execution of agent programs. The framework is written in Maude, and has a model of its physical environment to simulate faults on agent's sensors. Our framework differs on two points. First, the modular structure is very much apparent in the scenario of our framework: agents have their own module and interact through actions. This makes the update of an agent and the reuse of the same agent very easy. Second, our runtime has a by making the modularity of the framework structural

1.3 Running example

We introduce, through an example, some intuitive concepts that we will formalize later. We consider a cyber-physical system as a set of interacting processes. Whether a process consists of a physical phenomenon (sun rising, electro-chemical reaction, etc.) or a cyber phenomenon (computation of a function, message exchanges, etc.), it exhibits an externally observable behavior resulting from some internal non-visible actions. Instead of a unified way to describe internals of cyber and physical processes, we propose in Section 2.1 a uniform description of what we can externally observe of their behavior and interactions.

An *event* may describe something like *the sun-rise* or *the temperature reading of 5°C*. An event occurs at a point in time, yielding an event occurrence (e.g., the sun-rise event occurred at 6:28 am today), and the same event can occur repeatedly at different times (the sun-rise event occurs every day). Typically, multiple events may occur at “the same time" as measured within a measurement tolerance (e.g., the bird vacated the space at the same time as the bullet arrived there; the red car arrived at the middle of the intersection at the same time as the blue car did). We call a set of events that occur together at the same time an *observable.* A pair (O, t) of a set of observable events O together with its time-stamp t represents an *observation.* An observation (O, t) in fact consists of a set of event occurrences: occurrences of events in O at the same time t. We call an infinite sequence of observations a *Timed-Event Stream* (TES). A *behavior* is a set of TESs. A *component* is a behavior with an interface.

Consider two robot components, each interacting with its own local battery component, and sharing a field resource. The fact that the robots share the field through which they roam, forces them to somehow coordinate their (move) actions. Coordination is a set of constraints imposed on the otherwise possible observable behavior of components. In the case of our robots, if nothing else, at least physics prevents the two robots from occupying the same field space at the same time. More sophisticated coordination may be imposed (by the robots themselves or by some other external entity) to restrict the behavior of the robots and circumvent some undesirable outcomes, including hard constraints imposed by the physics of the field. The behaviors of components consist of timed-event streams, where events may include some measures of physical quantities. We give in the sequel a detailed description of three components, a robot (R), a battery (B), and a field (F), and of their interactions. We use the International System of Units to quantify physical values, with time in seconds (s), charging status in Watt hour (Wh), distance in meters (m), force in newtons (N), speed in meters per second ($\mathrm{m\,s^{-1}}$).

A *robot* component, with identifier R, has two kinds of events: a read event $(read(bat, R); b)$ that measures the level b of its battery or $(read(loc, R); l)$ that obtains its position l, and a move event $(move(R); (d, \alpha))$ when the robot moves in the direction d with energy α (in W). The TES in the Robot column in Table 1.1 shows a scenario where robot R reads its location and gets the value $(0; 0)$ at time 1s, then moves north with 20W at time 2s, reads its location and gets $(0; 1)$ at time 3s, and reads its battery value and gets 2000Wh at time 4s, ...

A *battery* component, with identifier B, has three kinds of events: a charge event $(charge(B); \eta_c)$, a discharge event $(discharge(B); \eta_d)$, and a read event $(read(B); s)$, where η_d and η_c are respectively the discharge and charge rates of the battery, and s is the current charge status. The TES in the Battery column in Table 1.1 shows a scenario where the battery discharged at a rate of 20W at time 2s, and reported its charge-level of 2000Wh at time 4s, ...

A *field* component, with identifier F, has two kinds of events: a position event $(loc(I); p)$ that obtains the position p of an object I, and a move event $(move(I); (d, F))$

Table 1.1: Each column displays a segment of a timed-event stream for a robot, a battery, and a field component, where observables are singleton events. For $t \in \mathbb{R}_+$, we use $R(t), B(t)$, and $F(t)$ to respectively denote the observable at time t for the TES in the Robot, the Battery, and the Field column. An explicit empty set is not mandatory if no event is observed.

	Robot (R)	Battery (B)	Field (F)
1.0s	$\{(\mathit{read}(\mathit{loc}, R); (0; 0))\}$		$\{(\mathit{loc}(I); (0; 0))\}$
2.0s	$\{(\mathit{move}(R); (N, 20\mathrm{W}))\}$	$\{(\mathit{discharge}(B); 20\mathrm{W})\}$	$\{(\mathit{move}(I); (N, 40\mathrm{N}))\}$
3.0s	$\{(\mathit{read}(\mathit{loc}, R); (0; 1))\}$		$\{(\mathit{loc}(I); (0; 1))\}$
4.0s	$\{(\mathit{read}(\mathit{bat}, R); 2000\mathrm{Wh})\}$	$\{(\mathit{read}(B); 2000\mathrm{Wh})\}$	
...	...	...	...

	Robot-Battery-Field
1.0s	$R(1) \cup F(1)$
2.0s	$R(2) \cup B(2) \cup F(2)$
3.0s	$R(3) \cup F(3)$
4.0s	$R(4) \cup B(4)$
...	...

of the object I in the direction d with traction force F (in N). The TES in the Field column in Table 1.1 shows a scenario where the field has the object I at location $(0; 0)$ at time 1s, then the object I moves in the north direction with a traction force of 40N at time 2s, subsequently to which the object I is at location $(0; 1)$ at time 3s, ...

When components interact with each other, in a shared environment, behaviors in their composition must also compose with a behavior of the environment. For instance, a battery component may constrain how many amperes it delivers, and therefore restrict the speed of the robot that interacts with it. We specify interaction explicitly as an exogenous binary operation that constrains the composable behaviors of its operand components.

The *robot-battery* interaction imposes that a move event in the behavior of a robot coincides with a discharge event in the behavior of the robot's battery, such that the discharge rate of the battery is proportional to the energy needed by the robot. The physicality of the battery prevents the robot from moving if the energy level of the battery is not sufficient (i.e., such an anomalous TES would not exist in the battery's behavior, and therefore cannot compose with a robot's behavior). Moreover, a read event in the behavior of a robot component should also coincide with a read event in the behavior of its corresponding battery component, such that the two events contain the same charge value.

The *robot-field* interaction imposes that a move event in the behavior of a robot coincides with a move event of an object on the field, such that the traction force on the field is proportional to the energy that the robot puts in the move. A read event in the behavior of a robot coincides with a position event of the corresponding robot object on the field, such that the two events contain the same position value. Additional interaction constraints may be imposed by the physics of the field. For instance, the constraint "no two robots can be observed at the same location" would rule out every behavior where the two robots are observed at the same location.

A TES for the composite Robot-Battery-Field system collects, in sequence, all observations from a TES in a Robot, a Battery, and a Field component behavior, such that at any moment the interaction constraints are satisfied. The column Robot-Battery-Field in Table 1.1 displays the first elements of such a TES.