An algebra for interaction of cyber-physical components

Lion, B.

Chapter 2

A semantic model for interacting cyber-physical systems

Cyber-physical systems often describe systems in which a program (cyber) has to regulate and control a physical quantity. For instance, consider a heating system equipped with sensors and a controller. The controller has as objective to maintain the temperature of a room within some bounds. The controller (cyber) frequently reads the temperature sensors located in the room, and takes decision as to turn the heater on or off. The decision made by the controller changes the dynamic of the temperature profile, and eventually the next readings of the sensors. A similar structure is observed if one wants to direct autonomously a car on a field. The car has some position and energy sensors, and actuates its wheels depending on the value that its controller reads. The decision made by the car changes the location of the car on the field and modifies the remaining energy in its battery.

The two examples above are commonly considered to belong to the field of *control theory*, i.e., the design of control algorithms that monitor observable measures to maintain some invariants. The design of a control algorithm is directly subject to the physical system with which it interacts. As the physical system becomes more complex, finding a suitable model becomes challenging as well. For instance, if the room has several heaters and sensors distributed over the space, the decision taken at one heater may interfere with the decision taken at the other heater. Coordination is

then necessary between controllers. Similarly, in the case where two robots move on the same shared field, the decision of one robot may directly depend on the decision of the other robot to move.

Our model, introduced in Section 2.1, builds on top of existing *control theory* to define cyber-physical systems as a composition of its parts. Similar approaches are taken in software design [3, 56, 34], where a complex monolithic software is broken into subparts that interact. Doing so requires a model to specify explicitly the interaction that occurs in between each parts. We use *components* to refer to the parts of a cyber-physical system, and defines several *product* operations over such components to express their interactions. Intuitively, a component encapsulates both a cyber or a physical process.

Composition is an important feature of a specification language, as it enables the design of a complex system in terms of a product of its parts. Decomposition is equally important in order to reason about structural properties. Usually, however, a system can be decomposed in more than one way, each optimizing for different criteria. In Section 2.2, we extend our model to reason about decomposition. Components compose using a family of algebraic products, and decompose, under some conditions, given a corresponding family of division operators. We use division to specify invariants of a system of components, and to model desirable updates. We apply our framework to design a cyber-physical system consisting of robots moving on a shared field, and identify desirable updates using our division operator

As a theoretical application of our model, we study in Section 2.3 the 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 consider the class of components for which occurrences of events are independent of the precise value of the time at which they happen, but depend on the past or future occurrences of other events. We give conditions for a linearization to be valid, which intuitively preserves the integrity of the behaviors after linearization, and give two instances of valid linearizations.

2.1 Algebra of Components

The definition of components in this section is similar to the one defined in [3, 56]. Intuitively, a component denotes a set of (infinite) sequences of observations. Whether it is a cyber process or a physical process, our notion of component captures all of its possible sequences of observations.

A model of interaction emerges naturally from our component model by relating observation of events from one component to observation of events from another component. Moreover, we give a construction to lift constraints on observations to constraints on infinite sequences of observations, and ultimately define, from those interaction constraints, algebraic operations on components.

2.1.1 Notations

An *event* is a simplex (the most primitive form of an) observable element. An event may or may not have internal structure. For instance, the successive ticks of a clock are occurrences of a tick event that has no internal structure; successive readings of a thermometer, on the other hand, constitute occurrences of a temperature-reading event, each of which has the internal structure of a name-value pair. Similarly, we can consider successive transmissions by a mobile sensor as occurrences of a structured event, each instance of which includes geolocation coordinates, barometric pressure, temperature, humidity, etc. Regardless of whether or not events have internal structures, in the sequel, we regard events as uninterpreted simplex observable elements.

Notation 1 (Events)**.** *We use* $\mathbb{E}$ *to denote the universal set of events.*

An *observable* is a set of event occurrences that happen together and an *observation* is a pair (O, t) of an observable O and a time-stamp $t \in \mathbb{R}_+$.[1] An observation (O, t) represents an act of atomically observing occurrences of events in O at time t. Atomicity, in its general form, consists of two properties: all events in the set must occur together, and no *interfering* event can occur in between any two events from the set. The second clause is in general formalized by a dependence relation (see Section 2.3). In the case of an observation, atomically observing occurrences of events in O at time t means there exists a small $\epsilon \in \mathbb{R}_+$ such that during the time interval $[t - \epsilon, t + \epsilon]$:

1. every event $e \in O$ is observed exactly once[2], and
2. no event $e \notin O$ is observed.

In the absence of a specified dependence relation, we make the safe, conservative assumption that all events depend on each other. Therefore, the atomicity of an

[1] Any totally ordered dense set would be suitable as the domain for time (e.g., positive rationals $\mathbb{Q}_+$). For simplicity, we use $\mathbb{R}_+$, the set of real numbers $r \geq 0$ for this purpose.

[2] A finer time granularity, i.e., a smaller ϵ, may reveal some ordering relation on the set of events that occur in the same set of observation.

observation, defined above, assumes the dependence relation to be total, i.e., all events are dependent, and no event can interleave within an observation.

We write $\langle s_0, s_1, ..., s_{n-1} \rangle$ to denote a *finite sequence of size n* of elements over an arbitrary set S, where $s_i \in S$ for $0 \leq i \leq n-1$. The set of all finite sequences of elements in S is denoted as S^*. A *stream* over a domain S is a function $\sigma : \mathbb{N} \to S$.[3] We use $\sigma(i)$ to represent the $i+1^{st}$ element of σ, and given a finite sequence $s = \langle s_0, ..., s_{n-1} \rangle$, we write $s \cdot \sigma$ to denote the stream $\tau \in \mathbb{N} \to S$ such that $\tau(i) = s_i$ for $0 \leq i \leq n-1$ and $\tau(i) = \sigma(i-n)$ for $n \leq i$. We use $\sigma^{(n)}$ to denote the *n-th* derivative of σ, such that $\sigma^{(n)}(i) = \sigma(i+n)$ for all $i \in \mathbb{N}$. We use σ' as an abbreviation for the first derivative of the stream σ, i.e., $\sigma' = \sigma^{(1)}$. We use $\mathcal{P}(X)$ to denote the power set of X.

A *Timed-Event Stream (TES)* over a set of events E and a set of time-stamps $\mathbb{R}_+$ is a stream $\sigma \in \mathbb{N} \to (\mathcal{P}(E) \times \mathbb{R}_+)$ where, for every $i \in \mathbb{N}$, let $\sigma(i) = (O_i, t_i)$ and:

1. $t_i < t_{i+1}$, [i.e., time monotonically increases] and
2. for every $n \in \mathbb{N}$, there exists $k \in \mathbb{N}$ such that $t_k > n$ [i.e., time is non-Zeno progressive].

Notation 2 (Time stream)**.** *We use $OS(\mathbb{R}_+)$ to refer to the set of all monotonically increasing and non-Zeno infinite sequences of elements in $\mathbb{R}_+$.*

Notation 3 (Timed-Event Stream)**.** *We use $TES(E)$ to denote the set of all TESs whose observables are subsets of the event set E with elements in $\mathbb{R}_+$ as their time-stamps.*

Given a sequence $\sigma \in TES(E)$ with $\sigma(i) = (O_i, t_i)$ for $i \in \mathbb{N}$, we use the projections $\mathsf{pr}_1(\sigma) \in \mathbb{N} \to \mathcal{P}(E)$ and $\mathsf{pr}_2(\sigma) \in OS(\mathbb{R}_+)$ to denote respectively the sequence of observables where $\mathsf{pr}_1(\sigma)(i) = O_i$ and the sequence of time stamps where $\mathsf{pr}_2(\sigma)(i) = t_i$.

Notation 4 (Observable time)**.** *For $\sigma \in TES(E)$ and $t \in \mathbb{R}_+$, we use $\sigma(t)$ to denote the observable O in σ if there exists $i \in \mathbb{N}$ with $\sigma(i) = (O, t)$, and $\emptyset$ otherwise. We write $\Theta(\sigma)$ for the set of all $t \in \mathbb{R}_+$ such that there exists $i \in \mathbb{N}$ with $\sigma(i) = (O_i, t)$ with $O_i \subseteq E$.*

Note that, for $t \in \mathbb{R}_+$ where $\sigma(t) = \emptyset$, the meaning of $\sigma(t)$ is ambiguous as it may mean either $t \notin \Theta(\sigma)$, or there exists an $i \in N$ such that $\sigma(i) = (\emptyset, t)$. The ambiguity is resolved by checking if $t \in \Theta(\sigma)$.

[3] The set $\mathbb{N}$ denotes the set of natural numbers $n \geq 0$.

Notation 5 (Pair derivative)**.** *For a pair (σ, τ) of TESs, we use $(\sigma, \tau)'$ to denote the new pair of TESs for which the observation(s) with the smallest time stamp has been dropped, i.e., $(\sigma, \tau)' = (\sigma^{(x)}, \tau^{(y)})$ with x (resp. y) is 1 if $\mathsf{pr}_2(\sigma)(0) \leq \mathsf{pr}_2(\tau)(0)$ (resp. $\mathsf{pr}_2(\tau)(0) \leq \mathsf{pr}_2(\sigma)(0)$) and 0 otherwise.*

2.1.2 Components

The design of complex systems becomes simpler if such systems can be decomposed into smaller sub-systems that interact with each other. In order to simplify the design of cyber-physical systems, we abstract from the internal details of both cyber and physical processes, to expose a uniform semantic model. As a first class entity, a component encapsulates a behavior (set of TESs) and an interface (set of events).

Like existing semantic models, such as time-data streams [3], time signal [87], or discrete clock [34], we use a dense model of time. However, we allow for arbitrary but finite interleavings of observations. In addition, our structure of an observation imposes atomicity of event occurrences within an observation. These distinctions mean that for every $\sigma(i) = (O, t), i \geq 0$ of a $\sigma \in TES(E)$: (1) O is finite; and (2) there exists a real number $\epsilon > 0$ such that in the open interval $(t - \epsilon, t + \epsilon)$ no event $e \notin O$ occurs, and every event $e \in O$ occurs exactly once. Such a constraint abstracts from the precise timing of the occurrence of each event in the set O, and turns an observation into an all-or-nothing transaction.

Definition 1 (Component)**.** *A component is a tuple $C = (E, L)$ where $E \subseteq \mathbb{E}$ is a set of events, and $L \subseteq TES(E)$ is a set of TESs. We call E the* interface *and L the externally observable* behavior *of C.*

As a shorthand notation, given component $C = (E, L)$, we use $\sigma : C$ for $\sigma \in L$.

In contrast with other component models where observables range over the same universal set of events, therefore making component overly specified, our model encapsulates the set of observable events of a component in its interface. Thus, a component *cannot observe* an event that is not in its interface. Moreover, Definition 1 makes no distinction between cyber and physical components. We use the following examples to describe some cyber and physical aspects of components.

Example 1. *Consider a set of two events $E = \{0, 1\}$, and restrict our observations to $\{1\}$ and $\{0\}$. A component whose behavior contains TESs with alternating observations*

of $\{1\}$ *and* $\{0\}$ *is defined by the tuple* (E, L) *where*

$$L = \{\sigma \in TES(E) \mid \forall i \in \mathbb{N}. \quad (\mathsf{pr}_1(\sigma)(i) = \{0\} \wedge \mathsf{pr}_1(\sigma)(i+1) = \{1\}) \vee (\mathsf{pr}_1(\sigma)(i) = \{1\} \wedge \mathsf{pr}_1(\sigma)(i+1) = \{0\})\}$$

Note that this component is oblivious to time, and any stream of monotonically increasing non-Zeno real numbers would serve as a valid stream of time stamps for any such sequence of observations. ■

Example 2. *Consider a component encapsulating a continuous function* $f : (D_0 \times \mathbb{R}_+) \to D$*, where* D_0 *is a set of initial values, and* D *is the codomain of values for* f*. Such a function can describe the evolution of a physical system over time, where* $f(d_0, t) = d$ *means that at time* t *the state of the system is described by the value* $d \in D$ *if initialized with* d_0*. We define the set of all events for this component as the range of function* f *given an initial parameter* $d_0 \in D_0$*. The component is then defined as the pair* (D, L_f) *such that:*

$$L_f = \{\sigma \in TES(D) \mid \exists d_0 \in D_0.\ \forall i \in \mathbb{N}.\ \mathsf{pr}_1(\sigma)(i) = \{f(d_0, \mathsf{pr}_2(\sigma)(i))\}\}$$

Observe that the behavior of this component contains all possible discrete samplings of the function f *at monotonically increasing and non-Zeno sequences of time stamp. Different instances of* f *would account for various cyber and physical aspects of components. The function* f *could be specified using, for instance, hybrid automtata or differential equations.* ■

Components are declarative entities that may denote either the behavior of a specification, or the behavior of an implementation. The usual relation between the behavior of a program and the property of such program constitutes a refinement relation.

Definition 2 (Refinement)**.** *A component* B *is a refinement of component* A*, written as* $B \sqsubseteq A$*, if and only if* $E_B \subseteq E_A$ *and* $L_B \subseteq L_A$*.*

Lemma 1. *The relation* $\sqsubseteq$ *is a partial order on components.*

Proof. Follows from reflexivity, antisymmetry, and transitivity of set inclusion. □

An alternative to refinement is *containment*. The containment relation makes use of a point-wise inclusion relation on observations of two TESs. The containment relation on components requires that every TES in the behavior of one is point-wise contained in a TES from the behavior of the other.

Definition 3 (Containment)**.** *A TES σ is contained in a TES τ, written as $\sigma \leq \tau$, if and only if, for all $i \in \mathbb{N}$, $\mathsf{pr}_1(\sigma)(i) \subseteq \mathsf{pr}_1(\tau)(i)$ and $\mathsf{pr}_2(\sigma) = \mathsf{pr}_2(\tau)$.*

Remark 1. *The restriction in Lemma 2 to consider components with no internal self containments between distinct TESs is necessary for having $\leq$ as a partial order. Consider for instance the component A with only two TESs in its behavior, $\sigma : A$ and $\tau : A$ where $\mathsf{pr}_1(\sigma) = (\{a, b\})^\omega$ and $\mathsf{pr}_1(\tau) = (\{a\})^\omega$ and $\mathsf{pr}_2(\sigma) = \mathsf{pr}_2(\tau)$. Let B be a component with a singleton behavior $\delta : B$ such that $\mathsf{pr}_1(\delta) = (\{a, b\})^\omega$ and $\mathsf{pr}_2(\delta) = \mathsf{pr}_2(\sigma)$. Then, $A \leq B$, and $B \leq A$, but $A \neq B$.*

We extend the containment relation to components: a component $A = (E_A, L_A)$ is contained in a component $B = (E_B, L_B)$, written $A \leq B$, if and only if $E_A \subseteq E_B$, and for every $\sigma \in L_A$, there exists a $\tau \in L_B$ such that $\sigma \leq \tau$.

Lemma 2. *The relation $\leq$ is a pre-order over arbitrary set of components. Moreover, $\leq$ is a partial-order over the set of components $\mathcal{C}$ if, for all components $A \in \mathcal{C}$ and for any two TESs $\sigma : A$ and $\tau : A$, $(\sigma \leq \tau \wedge \tau \leq \sigma) \implies \sigma = \tau$.*

Proof. Let $A = (E_A, L_A)$, $B = (E_B, L_B)$, and $C = (E_C, L_C)$ be three components. We show that $\leq$ is reflexive, transitive, and antisymmetric for any set $\mathcal{C}$ that satisfies the above condition:

1. reflexivity: $A \leq A$ holds.
2. transitivity. Let $A \leq B$ and $B \leq C$. Then, for all $\sigma : A$, there exists $\tau : B$ such that $\sigma \leq \tau$, and for all $\tau : B$, there exists $\delta : C$ such that $\tau \leq \delta$. Then, we conclude that for all $\sigma : A$, there exists $\delta : C$ such that $\sigma \leq \delta$ and $A \leq C$.
3. antisymmetric. We suppose that A and B are elements of the set $\mathcal{C}$. If $A \leq B$ and $B \leq A$, then for all $\sigma : B$, there exists $\tau : A$ such that $\sigma \leq \tau$. As well, for any $\tau : A$, there exists $\sigma : B$ such that $\tau \leq \sigma$. Thus, for any $\sigma : B$, there exists $\tau : A$ and $\delta : B$ with $\sigma \leq \tau \leq \delta$. Given the assumption of A and B, we can conclude that $\sigma = \tau = \delta$. Similarly, we show that $L_A \subseteq L_B$, and that $A = B$.

□

2.1.3 Composition

A complex system typically consists of multiple components that interact with each other. The running example in Section 1.3 shows three components, a *robot*, a *bat*, and a *field*, where, for instance, a move observable of a robot must coincide with an

accommodating move observable of the field and a discharge observable of its battery. The design challenge is to faithfully represent the interactions among involved components, while keeping the description modular, i.e., specify the robot, the battery, and the field as separate, independent, but interacting components. For that purpose, we capture in an interaction signature the type of the interaction between a pair of components, and we define a family of binary products acting on components, each parametrized with an interaction signature. As a result, the product of two components, under a given interaction signature, returns a new component whose behavior reflects that the two operand components joint behavior is constrained according to the interaction signature. Such construction opens possibilities for modular reasoning both about the interaction among components and about their resulting composite behavior.

An interaction signature consists of two elements: a composability relation and a composition function. The composability relation specifies which pairs[4] of TESs are allowed to compose, and the composition function constructs a new TES out of a pair TESs. The condition for two TESs to be composable may depend on an external context. For instance, the observation of event a at time t in a TES may conflict with the observation of event b at that same time t in another TES in a context where the latter could have observed a as well. To capture this notion, we generalized the notion of a composability relation to take as parameter a pair of carrier sets of events that acts as a context of alternative events for the pair of TESs. Then, when we write $(\sigma, \tau) \in R(E_1, E_2)$, we mean that σ and τ are composable under the composability relation R given their respective context E_1 and E_2.

Definition 4 (Composability relation on TESs)**.** *A composability relation is a parametrized relation R such that for all $E_1, E_2 \subseteq \mathbb{E}$, we have $R(E_1, E_2) \subseteq TES(E_1) \times TES(E_2)$.*

Definition 5 (Symmetry)**.** *A parametrized relation Q is* symmetric *if, for all (x_1, x_2) and for all (X_1, X_2): $(x_1, x_2) \in Q(X_1, X_2) \iff (x_2, x_1) \in Q(X_2, X_1)$.*

A composability relation on TESs serves as a necessary constraint for two TESs to compose. We define *composition* of TESs as the act of forming a new TES out of two TESs.

Definition 6. *A composition function $\oplus$ on TES is a function $\oplus : TES(\mathbb{E}) \times TES(\mathbb{E}) \to TES(\mathbb{E})$.*

[4]Non-binary relations may also be considered, i.e., constraints imposed on more than two components.

In order to simplify the development of the theory of components, we group a pair of a composability relation and a composition function into an *interaction signature*.

Definition 7. *An interaction signature $\Sigma = (R, \oplus)$ is a pair of a composability relation R and a composition function $\oplus$.*

Example 3 (Union of TESs)**.** *The operation $\cup$ forms the interleaved union of observables occurring in a pair of TESs, i.e., for two TESs σ and τ, we define $\sigma \cup \tau$ to be the TES such that $\Theta(\sigma \cup \tau) = \Theta(\sigma) \cup \Theta(\tau)$ and $(\sigma \cup \tau)(t) = \sigma(t) \cup \tau(t)$ for all $t \in \Theta(\sigma) \cup \Theta(\tau)$.* ■

The following examples present some useful interaction signatures for composition of TESs that, e.g., enforce synchronization or mutual exclusion of observables.

Example 4 (Synchronous interaction)**.** *In this example, we define the synchronous interaction signature $\Sigma_{sync} = (R_{sync}, \cup)$. In a cyber-physical system, the action (of a cyber system) and the reaction (of a physical system) co-exist simultaneously in the same observation, and are therefore synchronous.*

Then, $R_{sync}(E_1, E_2)$ relates pairs of TESs such that all shared events occur at the same time in both TESs, i.e., $(\sigma, \tau) \in R_{sync}(E_1, E_2)$ if and only if, for all time stamps $t \in \mathbb{R}_+$, $\sigma(t) \cap E_2 = \tau(t) \cap E_1$. A synchronous interaction signature Σ_{sync} filters pairs of TESs that satisfy the R_{sync} relation and merges composable pairs of observations. ■

Example 5 (Asynchronous interaction)**.** *In this example, we define the asynchronous interaction signature $\Sigma_{async} = (R_{async}, \cup)$. Typically, the asynchronous interaction signature prevents the same event to occur at the same time in a pair of TESs.*

Then, $R_{async}(E_1, E_2)$ relates pair of TESs such that all shared event between E_1 and E_2 occur at different time, i.e., $(\sigma, \tau) \in R_{async}(E_1, E_2)$ if and only if $\sigma(t) \cap \tau(t) = \emptyset$ for all $t \in \Theta(\sigma) \cup \Theta(\tau)$. An asynchronous interaction signature $\Sigma = (R_{async}, \cup)$ filters pairs of TESs that satisfy the R_{async} relation and merges composable pairs. ■

Example 6 (Free interaction)**.** *A free interaction signature, $\Sigma_{free} = (R_{free}, \cup)$, uses R_{free} for the most permissive composability relation on TESs such that, for any $E_1, E_2 \subseteq \mathbb{E}$ and any $\sigma \in TES(E_1)$ and $\tau \in TES(E_2)$, we have $(\sigma, \tau) \in R_{free}(E_1, E_2)$.* ■

We define a binary product operation on components, parametrized by an interaction signature. Intuitively, the newly formed component describes, by its behavior, the evolution of the joint system under the constraint that the interactions in the system satisfy the composability relation. Formally, the product operation returns another

component, whose set of events is the union of sets of events of its operands, and its behavior is obtained by composing all pairs of TESs in the behavior of its operands deemed composable by the composability relation.

Definition 8 (Product)**.** *Let $\Sigma = (R, \oplus)$ be an interaction signature, and $C_i = (E_i, L_i)$, $i \in \{1, 2\}$, two components. The product of C_1 and C_2, under Σ, denoted as $C_1 \times_\Sigma C_2$, is the component (E, L) where $E = E_1 \cup E_2$ and L is defined by*

$$L = \{\sigma_1 \oplus \sigma_2 \mid \sigma_1 \in L_1,\ \sigma_2 \in L_2,\ (\sigma_1, \sigma_2) \in R(E_1, E_2)\}$$

The following examples define several products on components given the interaction signatures introduced in Example 4, 5, and 6.

Example 7 (Synchronous product)**.** *The behavior of component $C_1 \times_{(R_{sync}, \oplus)} C_2$ contains TESs obtained from the composition under $\oplus$ of every pair $\sigma_1 \in L_1$ and $\sigma_2 \in L_2$ of TESs that are related by the synchronous composability relation R_{sync} (see Example 4) which excludes all event occurrences that do not synchronize. In the case where $\oplus = \cup$, we write $\bowtie = \times_{(R_{sync}, \cup)}$, and we call the operator $\bowtie$ the join operator.* ■

Example 8 (Asynchronous product)**.** *The behavior of component $C_1 \times_{(R_{async}, \oplus)} C_2$ contains TESs resulting from the composition under $\oplus$ of every pair $\sigma_1 \in L_1$ and $\sigma_2 \in L_2$ of TESs that are related by the mutual exclusion composability relation R_{async} (see Example 5) which may exclude some simultaneous event occurrences. In the case where $\oplus = \cup$, we write $\between = \times_{(R_{async}, \cup)}$.* ■

Example 9 (Free product)**.** *The behavior of component $C_1 \times_{(R_{free}, \oplus)} C_2$ contains every TES obtained from the composition under $\oplus$ of every pair $\sigma_1 \in L_1$ and $\sigma_2 \in L_2$ of TESs. This product does not impose any constraint on event occurrences of its operands (see Example 6).* ■

Example 10. *Consider a suitable interaction signature Σ that captures the interactions between a robot R and its field F, such that the expression $R \times_\Sigma F$ represents the resulting system. For instance, Σ may force every observable move of the robot to synchronize with a displacement of the robot on the field F, and every read observable of the robot with a location displayed by the field. In the case of two interacting robots roaming on the same field, one would like to build the resulting system compositionally as an expression of the form $(R_1 \times_{\Sigma_1} F_1) \times_{\Sigma_3} (R_2 \times_{\Sigma_2} F_2)$, where each of the Σ_i locally captures the interaction between its respective component. Note that, for instance, Σ_3 may enforce that the two fields F_1 and F_2 exclude the joint observations of R_1 and R_2 to be at the same location.* ■

The product of two components indirectly depends on the interface of its operands, since its composability relation does so. Therefore, it is *a priori* not certain that algebraic properties such as commutativity or associativity hold for such user defined products. Algebraic properties are important when designing a complex system in order to find equivalent and sometimes simpler expressions. Lemma 3 relates properties of a parametrized product with properties of its parameter, i.e., properties of the interaction signature. Intuitively, the first item of Lemma 3 considers interaction signatures that yield symmetric operations. As a result, the order of which components appear in the product parametrized by such signatures is irrelevant. The second item shows conditions on interaction signatures that allow flattening of nested products: the product of A with $B \times_\Sigma C$ becomes equivalent to the product of $A \times_\Sigma B$ with C. When an interaction signature satisfies both algebraic properties, the resulting product acts as an n-ary top level operator on a multiset of components. For instance, the synchronous interaction signature of Example 4 is one such top level n-ary operator.[5]

Lemma 3. *Let $\Sigma = (R, \oplus)$ be an interaction signature. Then:*

- *if R is symmetric, then $\times_\Sigma$ is* commutative *if and only if $\sigma_1 \oplus \sigma_2 = \sigma_2 \oplus \sigma_1$ for all $(\sigma_1, \sigma_2) \in R$;*

- *if R is such that, for all $E_1, E_2, E_3 \subseteq \mathbb{E}$,*

$$(\sigma_1, \sigma_2 \oplus \sigma_3) \in R(E_1, E_2 \cup E_3) \wedge (\sigma_2, \sigma_3) \in R(E_2, E_3) \iff$$
$$(\sigma_1, \sigma_2) \in R(E_1, E_2) \wedge (\sigma_1 \oplus \sigma_2, \sigma_3) \in R(E_1 \cup E_2, E_3)$$

  then $\times_\Sigma$ is associative *if and only if $\sigma_1 \oplus (\sigma_2 \oplus \sigma_3) = (\sigma_1 \oplus \sigma_2) \oplus \sigma_3$ for all $(\sigma_1, \sigma_2 \oplus \sigma_3) \in R(E_1, E_2 \cup E_3)$ with $(\sigma_2, \sigma_3) \in R(E_2, E_3)$;*

- *if for all $E \subseteq \mathbb{E}$ and $\sigma, \tau \in TES(E)$, we have $(\sigma, \tau) \in R(E, E) \implies \sigma = \tau$, then $\times_\Sigma$ is* idempotent *if and only if $\sigma \oplus \sigma = \sigma$ for all $(\sigma, \sigma) \in R$.*

Proof. Commutativity. Let $C_1 = (E_1, L_1)$ and $C_2 = (E_2, L_2)$ be two components, and $\Sigma = (R, \oplus)$ be an interaction signature with R symmetric as in Definition 5. We write $C = (E, L) = C_1 \times_\Sigma C_2$ and $C' = (E', L') = C_2 \times_\Sigma C_1$. We first observe that $E = E_1 \cup E_2 = E'$. The condition for the product of two components to be

[5] Distributivity holds for some products. We leave the study of the conditions under which distributivity holds as future work.

commutative reduces to showing that $L = L'$, also equivalently written as:

$$\begin{aligned} L = L' \iff & \{\sigma_1 \oplus \sigma_2 \mid \sigma_1 \in L_1,\ \sigma_2 \in L_2,\ (\sigma_1, \sigma_2) \in R(E_1, E_2)\} \\ & = \{\sigma_2 \oplus \sigma_1 \mid \sigma_1 \in L_1,\ \sigma_2 \in L_2,\ (\sigma_2, \sigma_1) \in R(E_2, E_1)\} \end{aligned}$$

If $\sigma_1 \oplus \sigma_2 = \sigma_2 \oplus \sigma_1$ for $(\sigma_1, \sigma_2) \in R(E_1, E_2)$, then $L = L'$ and $\times_\Sigma$ is commutative.

Oppositely, if $L = L'$, we show that $\oplus$ is commutative. Let C_σ be the component $(E_\sigma, \{\sigma\})$ where $E_\sigma = \bigcup\{\sigma(i) \mid i \in \mathbb{N}\}$. Thus, for any $(\sigma_1, \sigma_2) \in R(E_1, E_2)$, $C_{\sigma_1} \times_\Sigma C_{\sigma_2} = (E_{\sigma_1} \cup E_{\sigma_2}, \{\sigma_1 \oplus \sigma_2\}$. A necessary condition for $\times_\Sigma$ to be commutative is that $\{\sigma_1 \oplus \sigma_2\} = \{\sigma_2 \oplus \sigma_1\}$, which imposes that $\sigma_1 \oplus \sigma_2 = \sigma_2 \oplus \sigma_1$.

Associativity. Let $(R, \oplus)$ be a pair of a composability relation and a composition function on TESs with R such that, for every $(\sigma_1, \sigma_2, \sigma_3) \in L_1 \times L_2 \times L_3$:

$$\begin{aligned} & (\sigma_1, \sigma_2) \in R(E_1, E_2) \wedge (\sigma_1 \oplus \sigma_2, \sigma_3) \in R(E_1 \cup E_2, E_3) \iff \\ & (\sigma_2, \sigma_3) \in R(E_2, E_3) \wedge (\sigma_1, \sigma_2 \oplus \sigma_3) \in R(E_1, E_2 \cup E_3) \end{aligned}$$

We consider three components $C_i = (E_i, L_i)$, with $i \in \{1, 2, 3\}$.

The set of events for component $((C_1 \times_\Sigma C_2) \times_\Sigma C_3)$ is the set $E_1 \cup E_2 \cup E_3$, which is equal to the set of events for component $(C_1 \times_\Sigma (C_2 \times_\Sigma C_3))$.

Let L' and L'' respectively be the behaviors of components $(C_1 \times_\Sigma C_2) \times_\Sigma C_3$ and $C_1 \times_\Sigma (C_2 \times_\Sigma C_3)$. If $\sigma_1 \oplus (\sigma_2 \oplus \sigma_3) = (\sigma_1 \oplus \sigma_2) \oplus \sigma_3$ for all $(\sigma_1, \sigma_2 \oplus \sigma_3) \in R(E_1, E_2 \cup E_3)$ with $(\sigma_2, \sigma_3) \in R(E_2, E_3)$, then $L' = L''$. We show some sufficient conditions for $L' = L''$, also written as

$$\begin{aligned} L' & = \{(\sigma_1 \oplus \sigma_2) \oplus \sigma_3 \mid \sigma_1 \in L_1,\ \sigma_2 \in L_2,\ \sigma_3 \in L_3.\ (\sigma_1, \sigma_2) \in R(E_1, E_2) \wedge \\ & \qquad\qquad (\sigma_1 \oplus \sigma_2, \sigma_3) \in R(E_1 \cup E_2, E_3)\} \\ & = \{(\sigma_1 \oplus \sigma_2) \oplus \sigma_3 \mid \sigma_1 \in L_1,\ \sigma_2 \in L_2,\ \sigma_3 \in L_3.\ (\sigma_2, \sigma_3) \in R(E_2, E_3) \wedge \\ & \qquad\qquad (\sigma_1, \sigma_2 \oplus \sigma_3) \in R(E_1, E_2 \cup E_3)\} \\ & = \{\sigma_1 \oplus (\sigma_2 \oplus \sigma_3) \mid \sigma_1 \in L_1,\ \sigma_2 \in L_2,\ \sigma_3 \in L_3.\ (\sigma_2, \sigma_3) \in R(E_2, E_3) \wedge \\ & \qquad\qquad (\sigma_1, \sigma_2 \oplus \sigma_3) \in R(E_1, E_2 \cup E_3)\} \\ & = L'' \end{aligned}$$

using the assumption on R for the first equality, and the assumption on $\oplus$ for the second equality.

Let $(\sigma_1, \sigma_2 \oplus \sigma_3) \in R(E_1, E_2 \cup E_3)$ with $(\sigma_2, \sigma_3) \in R(E_2, E_3)$, then $C_{\sigma_1} \times_\Sigma (C_{\sigma_2} \times_\Sigma C_{\sigma_3}) = (C_{\sigma_1} \times_\Sigma C_{\sigma_2}) \times_\Sigma C_{\sigma_3}$ which then implies that $\sigma_1 \oplus (\sigma_2 \oplus \sigma_3) = (\sigma_1 \oplus \sigma_2) \oplus \sigma_3$.

Idempotency. We show that if for all $E \subseteq \mathbb{E}$, and $\sigma, \tau \in TES(E)$, we have that

$(\sigma, \tau) \in R(E, E)$ implies $\sigma = \tau$, then $\times_\Sigma$ is idempotent if and only if $\sigma \oplus \sigma = \sigma$ for $(\sigma, \sigma) \in R(E, E)$. We first observe that, given a component $C = (E, L)$, the component $C \times_\Sigma C = (E, L')$ has the same set of events, E.

We show that $(\sigma_1, \sigma_2) \in R(E, E) \implies \sigma_1 = \sigma_2$ and $\oplus$ idempotent is a sufficient condition for having $L' = L$. Indeed,

$$\begin{aligned} L' &= \{\sigma_1 \oplus \sigma_2 \mid \sigma_1, \sigma_2 \in L,\ (\sigma_1, \sigma_2) \in R(E, E)\} \\ &= \{\sigma_1 \oplus \sigma_1 \mid \sigma_1 \in L\} \\ &= L \end{aligned}$$

Similar to the previous cases, if for all $E \subseteq \mathbb{E}$, and $\sigma, \tau \in TES(E)$, we have that $(\sigma, \tau) \in R(E, E)$ implies $\sigma = \tau$, then $\times_\Sigma$ is idempotent if and only if $\oplus$ is idempotent. Conversely, if $C_\sigma \times_\Sigma C_\sigma = C_\sigma$ and $(\sigma, \sigma) \in R(E, E)$, then $\sigma \oplus \sigma = \sigma$. □

Monotonicity shows that the inclusion of component's behavior is preserved by product. Let A and B be two components such that $B \sqsubseteq A$. Suppose that P is a component that models a property satisfied by component A and preserved under product with a component C, then P is satisfied by component B and component $B \times C$, by monotonicity. Note that the definition of monotonicity assumes $\times$ to be commutative. That assumption can be relaxed by defining left and right monotonicity.

Definition 9 (Monotonicity)**.** *Let $\times$ be a commutative product. Then, $\times$ is monotonic if and only if, for $B \sqsubseteq A$ and for any C, we have $B \times C \sqsubseteq A \times C$.*

Lemma 4 (Monotonicity of $\bowtie$)**.** *The product $\bowtie$ in Example 4 is monotonic.*

Proof. Let A, B, and C be three components, such that $B \sqsubseteq A$. Then, the interface of $B \bowtie C$ is $E_B \cup E_C$, which is included in $E_A \cup E_C$ the interface of $A \bowtie C$.

For any TES $\sigma : B \bowtie C$, there exist two TESs $\beta : B$ and $\delta : C$ such that (β, δ) are synchronous, and $\sigma = \beta \cup \delta$. Since for any $\beta : B$ we also have $\beta : A$, then σ is also an element of the behavior of $A \bowtie C$, and $B \bowtie C \sqsubseteq A \bowtie C$. □

The algebraic nature of our formalism allows the possibility to introduce other kinds of operations on components, such as division, as presented in Section 2.2. Intuitively, the operation of division is parametrized by an interaction signature Σ and follows two steps. First, the set of quotients of component A divided by component B is constructed as the set of all components C such that $A = B \times_\Sigma C$. Practically, every element in the set of quotients leads to the same composite behavior captured in A,

when composed with B under Σ. Then, one component is chosen from the set of quotients as the result of division.

2.1.4 A co-inductive construction

In this section, we show how local constraints on observations can be co-inductively *lifted* to global constraints on TESs. We get, as a result, a finite specification of some interaction signatures using simpler relations on observations. Moreover, we get a co-inductive proof mechanism to relate an interaction signature defined on TESs with an interaction signature lifted from constraints on observables, as shown in Lemma 7. Such construction gives, as well, an operational perspective on deriving an interaction signature as a step-wise constraint imposed on observables. Practically, the operational approach of the co-inductive definition is relevant when considering robots and their step-wise decision on their next observation.

The intuition for such construction is that, in some cases, the condition for two TESs to be composable depends only on a composability relation on observations. An example of composability constraint for a robot with its battery and a field enforces that each *move* event *discharges* the battery and *changes* the state of the field. As a result, every *move* event observed by the robot must coincide with a *discharge* event observed by the battery and a change of state observed by the field. The lifting of such composability relation on observations to a constraint on TESs is defined co-inductively. Finally, Lemma 10 gives weaker conditions for Lemma 3 to hold.

Definition 10 (Composability relation on observations)**.** *A composability relation on observations is a parametrized relation κ such that for all pairs $(E_1, E_2) \in \mathcal{P}(\mathbb{E})\times\mathcal{P}(\mathbb{E})$, we have $\kappa(E_1, E_2) \subseteq (\mathcal{P}(E_1) \times \mathbb{R}_+) \times (\mathcal{P}(E_2) \times \mathbb{R}_+)$.*

The following examples define locally on observations some relations analogous to those defined globally on TESs in Example 4 and Example 5.

Example 11. *We give two examples of composability relations on observations:*

- $((O_1, t_1), (O_2, t_2)) \in \kappa^{sync}(E_1, E_2)$ *if and only if every shared event always occurs at the same time, i.e., $t_1 < t_2$ implies $O_1 \cap E_2 = \emptyset$, and $t_2 < t_1$ implies $O_2 \cap E_1 = \emptyset$, and $t_2 = t_1$ implies $O_1 \cap E_2 = O_2 \cap E_1$;*

- $((O_1, t_1), (O_2, t_2)) \in \kappa^{async}(E_1, E_2)$ *if and only if no shared event occurs at the same time, i.e., $t_1 = t_2$ implies $O_1 \cap E_2 = \emptyset = O_2 \cap E_1$.* ■

For two composability relations κ_1, κ_2, their intersection or union, written $\kappa_1 \cap \kappa_2$ and $\kappa_1 \cup \kappa_2$ respectively, is defined, for any $E_1, E_2, E_3 \subseteq \mathbb{E}$, as $(\kappa_1 \cap \kappa_2)(E_1, E_2) = \kappa_1(E_1, E_2) \cap \kappa_2(E_1, E_2)$ and $(\kappa_1 \cup \kappa_2)(E_1, E_2) = \kappa_1(E_1, E_2) \cup \kappa_2(E_1, E_2)$.

Definition 11 (Lifting)**.** *Let κ be a composability relation on observations, and, for any $\mathcal{R} \subseteq TES(E_1) \times TES(E_2)$, let $\Phi_\kappa(E_1, E_2)(\mathcal{R}) \subseteq TES(E_1) \times TES(E_2)$ be such that:*

$$\Phi_\kappa(E_1, E_2)(\mathcal{R}) = \{(\tau_1, \tau_2) \mid (\tau_1(0), \tau_2(0)) \in \kappa(E_1, E_2) \wedge (\tau_1, \tau_2)' \in \mathcal{R})\}$$

The lifting *of κ on TESs, written $[\kappa]$, is the parametrized relation obtained by taking the greatest post fixed point of the function $\Phi_\kappa(E_1, E_2)$ for arbitrary pair $E_1, E_2 \subseteq \mathbb{E}$, i.e., the relation $[\kappa](E_1, E_2) = \bigcup_{\mathcal{R} \subseteq TES(E_1) \times TES(E_2)} \{\mathcal{R} \mid \mathcal{R} \subseteq \Phi_\kappa(E_1, E_2)(\mathcal{R})\}$.*

Lemma 5 (Correctness)**.** *For any $E_1, E_2 \subseteq \mathbb{E}$, the function $\Phi_\kappa(E_1, E_2)$ is monotone, and therefore has a greatest post fixed point.*

Proof. Let κ be a composability relation on observations, and let $E_1, E_2 \subseteq \mathbb{E}$. We recall that the function $\Phi_\kappa(E_1, E_2)$ is such that, for any $\mathcal{R} \subseteq TES(E_1) \times TES(E_2)$:

$$\Phi_\kappa(E_1, E_2)(\mathcal{R}) = \{(\tau_1, \tau_2) \mid \quad (\tau_1(0), \tau_2(0)) \in \kappa(E_1, E_2) \wedge (\tau_1, \tau_2)' \in \mathcal{R})\}$$

Let $\mathcal{R}_1, \mathcal{R}_2 \subseteq TES(E_1) \times TES(E_2)$ be such that $\mathcal{R}_1 \subseteq \mathcal{R}_2$. We show that $\Phi_\kappa(E_1, E_2)(\mathcal{R}_1) \subseteq \Phi_\kappa(E_1, E_2)(\mathcal{R}_2)$. For any $(\tau_1, \tau_2) \in TES(E_1) \times TES(E_2)$,

$$\begin{aligned}
(\tau_1, \tau_2) \in \Phi_\kappa(E_1, E_2)(\mathcal{R}_1) &\iff (\tau_1(0), \tau_2(0)) \in \kappa(E_1, E_2) \wedge (\tau_1, \tau_2)' \in \mathcal{R}_1) \\
&\implies (\tau_1(0), \tau_2(0)) \in \kappa(E_1, E_2) \wedge (\tau_1, \tau_2)' \in \mathcal{R}_2) \\
&\implies (\tau_1, \tau_2) \in \Phi_\kappa(E_1, E_2)(\mathcal{R}_2)
\end{aligned}$$

Therefore, $\mathcal{R}_1 \subseteq \mathcal{R}_2$ implies that $\Phi_\kappa(E_1, E_2)(\mathcal{R}_1) \subseteq \Phi_\kappa(E_1, E_2)(\mathcal{R}_2)$, and we conclude that $\Phi_\kappa(E_1, E_2)$ is monotonic. We use the Knaster-Tarski theorem, where the underlying lattice is the powerset of TESs with inclusion relation, for the existence of a greatest fixed point of the monotonic function $\Phi_\kappa(E_1, E_2)$ applying to that lattice [6]. Thus, $\Phi_\kappa(E_1, E_2)$ has a greatest fixed point defined as:

$$[\kappa](E_1, E_2) = \bigcup \{\mathcal{R} \mid \mathcal{R} \subseteq \Phi_\kappa(E_1, E_2)(\mathcal{R})\}$$

□

[6] The Knaster-Tarski theorem shows that the greatest fixed point of $\Phi_\kappa(E_1, E_2)$ is also the greatest post-fixed point of $\Phi_\kappa(E_1, E_2)$

Lemma 6. *If κ is a composability relation on observations, then the lifting $[\kappa]$ is a composability relation on TESs. Moreover, if κ is symmetric (as in Definition 5), then $[\kappa]$ is symmetric.*

Proof. We first note that, given a composability relation κ on observables, the lifting $[\kappa]$ is a composability relation on TESs. Indeed, for any pair of interfaces $E_1, E_2 \subseteq \mathbb{E}$, any $(\sigma, \tau) \in [\kappa](E_1, E_2)$ is a pair in $TES(E_1) \times TES(E_2)$.

If κ is symmetric (as in Definition 5), we show that $[\kappa]$ is also symmetric. Given a set $\mathcal{R} \subseteq TES(E_1) \times TES(E_2)$, we use the notation $\overline{\mathcal{R}}$ to denote the smallest set such that $(\sigma, \tau) \in \mathcal{R} \iff (\tau, \sigma) \in \overline{\mathcal{R}}$. Let $E_1, E_2 \subseteq \mathbb{E}$.

If κ is symmetric, then for $\mathcal{R} \subseteq TES(E_1) \times TES(E_2)$,

$$\begin{aligned}
\Phi_\kappa(E_1, E_2)(\mathcal{R}) &= \{(\tau_1, \tau_2) \mid \quad (\tau_1(0), \tau_2(0)) \in \kappa(E_1, E_2) \wedge (\tau_1, \tau_2)' \in \mathcal{R})\} \\
&= \{(\tau_1, \tau_2) \mid \quad (\tau_2(0), \tau_1(0)) \in \kappa(E_2, E_1) \wedge (\tau_1, \tau_2)' \in \mathcal{R})\} \\
&= \{(\tau_1, \tau_2) \mid \quad (\tau_2(0), \tau_1(0)) \in \kappa(E_2, E_1) \wedge (\tau_2, \tau_1)' \in \overline{\mathcal{R}})\} \\
&= \{(\tau_1, \tau_2) \mid \quad (\tau_2, \tau_1) \in \Phi_\kappa(E_2, E_1)(\overline{\mathcal{R}})\} \qquad (1)
\end{aligned}$$

which shows that $[\kappa]$ is symmetric since, for any $E_1, E_2 \subseteq \mathbb{E}$,

$$[\kappa](E_1, E_2) = \bigcup_{\mathcal{R} \subseteq TES(E_1) \times TES(E_2)} \{\mathcal{R} \mid \mathcal{R} \subseteq \Phi_\kappa(E_1, E_2)(\mathcal{R})\}$$

and

$$\begin{aligned}
(\sigma, \tau) \in [\kappa](E_1, E_2) &\iff \exists \mathcal{R}.\ (\sigma, \tau) \in \mathcal{R} \wedge \mathcal{R} \subseteq \Phi_\kappa(E_1, E_2)(\mathcal{R}) \\
&\iff \exists \overline{\mathcal{R}}.\ (\tau, \sigma) \in \overline{\mathcal{R}} \wedge \overline{\mathcal{R}} \subseteq \Phi_\kappa(E_2, E_1)(\overline{\mathcal{R}}) \\
&\iff (\tau, \sigma) \in [\kappa](E_2, E_1)
\end{aligned}$$

where the first equivalence is given by the fact that $[\kappa](E_1, E_2)$ is the greatest post fixed point of $\Phi_\kappa(E_1, E_2)$, the second equivalence is obtained from equality (1), and the third equivalence is given by the fact that $[\kappa](E_2, E_1)$ is the greatest post fixed point.

□

As a consequence of Lemma 6, any composability relation on observations gives rise to a composability relation on TESs. Lemma 7 relates (a)synchornous composability relations on TESs with (a)synchronous composability relations on observations.

Lemma 7. *Let R_{sync} and R_{async} be composability relations defined in Example 4 and Example 5, respectively. Let κ^{sync} be the relation on observations defined in Example 11. For all E_1 and E_2, $R_{sync}(E_1, E_2) = [\kappa^{sync}](E_1, E_2)$ and $R_{async}(E_1, E_2) =$*

$[\kappa^{async}](E_1, E_2)$. *When* $E_1 \cap E_2 = \emptyset$, *then* $R_{sync}(E_1, E_2) = R_{async}(E_1, E_2)$.

Proof. We proof the result for $R_{sync}(E_1, E_2) = [\kappa^{sync}](E_1, E_2)$ and similar reasoning can be applied for $R_{async}(E_1, E_2) = [\kappa^{async}](E_1, E_2)$. We first show $R_{sync}(E_1, E_2) \subseteq [\kappa^{sync}](E_1, E_2)$, which is equivalent to $R_{sync}(E_1, E_2) \subseteq \Phi_{\kappa^{sync}}(E_1, E_2)(R_{sync}(E_1, E_2))$. First, we observe that if $(\sigma, \tau) \in R_{sync}(E_1, E_2)$, then $(\sigma, \tau)' \in R_{sync}(E_1, E_2)$, as dropping the first observation(s) of σ and τ preserves the property imposed by R_{sync}. For all $(\sigma, \tau) \in R_{sync}(E_1, E_2)$, by definition of R_{sync} and κ^{sync}, we have that $(\sigma(0), \tau(0)) \in \kappa^{sync}(E_1, E_2)$. Since $\Phi_{\kappa^{sync}}(E_1, E_2)(R_{sync}(E_1, E_2))$ is equal to the set

$$\{(\sigma, \tau) \mid (\tau(0), \sigma(0)) \in \kappa^{sync}(E_1, E_2) \wedge (\sigma, \tau)' \in R_{sync}(E_1, E_2)\}$$

we conclude that $R_{sync}(E_1, E_2) \subseteq \Phi_{\kappa^{sync}}(E_1, E_2)(R_{sync}(E_1, E_2))$.

For the other inclusion, let first introduce $time((\sigma, \tau))$ as the smallest time stamp of the head of both streams σ and τ, i.e., $time((\sigma, \tau)) = \min(t_1, t_2)$ where $t_1 = \mathsf{pr}_2(\sigma)(0)$ and $t_2 = \mathsf{pr}_2(\tau)(0)$. For a pair $(\sigma, \tau) \in [\kappa^{sync}](E_1, E_2)$, we have:

$$\begin{aligned}
&(\sigma(0), \tau(0)) \in \kappa^{sync}(E_1, E_2) \wedge (\sigma, \tau)' \in [\kappa^{sync}](E_1, E_2) \\
&\implies \forall t < time((\sigma, \tau)').\ \sigma(t) \cap E_2 = \tau(t) \cap E_1 \wedge (\sigma, \tau)' \in [\kappa^{sync}](E_1, E_2) \\
&\implies \forall n \in \mathbb{N}. \forall t < time((\sigma, \tau)^{(n)}).\ \sigma(t) \cap E_2 = \tau(t) \cap E_1 \wedge \\
&\qquad\qquad (\sigma, \tau)^{(n)} \in [\kappa^{sync}](E_1, E_2) \\
&\implies \forall t.\ \sigma(t) \cap E_2 = \tau(t) \cap E_1 \\
&\implies (\sigma, \tau) \in R_{sync}
\end{aligned}$$

We conclude that $R_{sync} = [\kappa^{sync}]$. □

The composability relation κ relates observations, i.e., elements of $\mathcal{P}(E) \times \mathbb{R}_+$. We show in Definition 13 and Definition 14 how to define κ from a relation $\sqcap$ on sets of events, i.e., elements in $\mathcal{P}(E)$.

Definition 12 (Intersection)**.** *For any two components* $C_1 = (E_1, L_1)$ *and* $C_2 = (E_2, L_2)$*, we define the intersection* $C_1 \cap C_2$ *to be the component* $C_1 \times_{([\kappa^{sync}], [\cap])} C_2 = (E_1 \cup E_2, L)$ *with* κ^{sync} *defined in Example 11.* ■

For the following definitions, let $C_1 = (E_1, L_1)$ and $C_2 = (E_2, L_2)$ be two components, and $\oplus$ be a composition function on TESs. We use $\sqcap \subseteq \mathcal{P}(\mathbb{E}) \times \mathcal{P}(\mathbb{E})$ to range over relations on observables.

Definition 13 (Synchronous observations)**.** *The composability relations introduced in Example 11 can also be extended to synchronize pairs of distinct events. Two observations are synchronous under $\sqcap$ if, intuitively, the two following conditions hold:*

1. *every observable that can compose (under $\sqcap$) with another observable must occur simultaneously with one of its related observables; and*

2. *only an observable that does not compose (under $\sqcap$) with any other observable can happen before another observable, i.e., at a strictly lower time.*

To formalize the conditions above, we introduce the independence relation $ind_{\sqcap}(X, Y) = \forall x \subseteq X. \forall y \subseteq Y. (x, y) \notin \sqcap$.

The synchronous *composability relation on observations $\kappa_{\sqcap}^{sync}(E_1, E_2)$ is the smallest set such that, for all $O_1 \subseteq E_1$ and $O_2 \subseteq E_2$:*

- *if $(O_1, O_2) \in \sqcap \cup (\emptyset, \emptyset)$, then for all $(O_1', O_2') \in \mathcal{P}(E_1) \times \mathcal{P}(E_2)$ such that $ind_{\sqcap}(O_1', E_2)$ and $ind_{\sqcap}(E_1, O_2')$ and for all time stamps t, we have that $((O_1 \cup O_1', t), (O_2 \cup O_2', t)) \in \kappa_{\sqcap}^{sync}(E_1, E_2)$.*

- *if $ind_{\sqcap}(O_1, E_2)$, then for all $O_2' \subseteq E_2$ and for all $t_1 < t_2$, we have that the pair $((O_1, t_1), (O_2, t_2)) \in \kappa_{\sqcap}^{sync}(E_1, E_2)$. Reciprocally, if $ind_{\sqcap}(E_1, O_2)$ then for all $O_1' \subseteq E_1$ and $t_2 < t_1$, we have $((O_1, t_1), (O_2, t_2)) \in \kappa_{\sqcap}^{sync}(E_1, E_2)$;*

Example 12. *Although the relation in Definition 13 is a binary relation on observations, we show in this example how to synchronize multiple events transitively. For instance, consider three components, $A = (\{a\}, L_A)$, $B = (\{b\}, L_B)$, and $C = (\{c\}, L_C)$. Let $\sqcap$ be the smallest symmetric relation with $\{(\{a\}, \{b\}), (\{b\}, \{c\})\} \subseteq \sqcap$. Then, $\kappa_{\sqcap}^{sync}$ enforces every observable in A and C to occur at the same time as an observable in B. Let $\Sigma = ([\kappa_{\sqcap}^{sync}], \cup)$ with $\cup$ defined in Example 3. Observe that, in general, $(A \times_{\Sigma} B) \times_{\Sigma} C \neq (A \times_{\Sigma} C) \times_{\Sigma} B$. On the left hand side, the product of A and B synchronizes every occurrence of event a with an occurrence of event b, which results in observables of the form $\{a, b\}$ only (no interleaving is allowed by $\kappa_{\sqcap}^{sync}$). Since b and c are also related events, the composition with C leads to the component with observables $\{a, b, c\}$. On the right hand side, A and C have independent observables and their composition allows for every interleaving. The product with B, however, synchronizes every occurrence of event a or c with an occurrence of event b, which results in interleaving of observables $\{a, b\}$ and $\{c, b\}$. Finally, observe that the component $(A \times_{\Sigma} B) \times_{\Sigma} C$ transitively synchronizes occurrences of event a with occurrences of event c through occurrences of event b.* ■

The behavior of component $C_1 \times_{([\kappa_\sqcap^{sync}],\oplus)} C_2$ contains TESs obtained from the composition under $\oplus$ of every pair $\sigma_1 \in L_1$ and $\sigma_2 \in L_2$ of TESs that are related by the synchronous composability relation $[\kappa_\sqcap^{sync}]$ which, depending on $\sqcap$, excludes all event occurrences that do not synchronize.[7] Note that in the case where $\sqcap = \{(O, O) \mid O \subseteq E_1 \cup E_2 \setminus \emptyset\}$, then $\kappa_\sqcap^{sync} = \kappa^{sync}$.

Definition 14 (Mutual exclusion)**.** *Let $\sqcap \subseteq \mathcal{P}(\mathbb{E})^2$ be a relation on observables. We define two observations to be mutually exclusive under the relation $\sqcap$ if no pair of observables in $\sqcap$ can be observed at the same time. The mutually exclusive composability relation $\kappa_\sqcap^{excl}$ on observations allows the composition of two observations (O_1, t_1) and (O_2, t_2), i.e., $((O_1, t_1), (O_2, t_2)) \in \kappa_\sqcap^{excl}(E_1, E_2)$, if and only if $t_1 = t_2 \implies \neg(O_1 \sqcap O_2)$.*

Example 13. *Following Example 10, we introduce an interaction signature that composes two robot-field subsystems while excluding the possibility for the robots to both observe the same location on their fields. We define $\sqcap = \{(\{(loc(R_1); l)\}, \{(loc(R_2); l)\}) \mid l \in [0; 20] \times [0; 20]\}$ as the set of pairs of observables containing, for both robots R_1 and R_2, an event that displays the same location as the other robot. Let $\Sigma = ([\kappa_\sqcap^{excl}], \cup)$ with $\cup$ defined in Example 3. Then, the product of the two subsystems, using the interaction signature Σ, excludes the possibility for the two robots to observe the same location at the same time. Strictly speaking, the exclusion imposed by the interaction signature Σ does not imply that the two robots can not* effectively *be on the same physical location. We show in Section 2.1.6 how, combined with hyper-properties, such interaction signature may imply a safety property.* ■

The behavior of component $C_1 \times_{([\kappa_\sqcap^{excl}],\oplus)} C_2$ contains TESs resulting from the composition under $\oplus$ of every pair $\sigma_1 \in L_1$ and $\sigma_2 \in L_2$ of TESs that are related by the mutual exclusion composability relation $[\kappa_\sqcap^{excl}]$ which, depending on $\sqcap$, may exclude some simultaneous event occurrences.

We prove in Lemma 8 that the lifting of composability relations distributes across the intersection.[8]

Lemma 8. *For all composability relations κ_1, κ_2 and interfaces E_1, E_2:*

$$[\kappa_1 \cap \kappa_2](E_1, E_2) = [\kappa_1](E_1, E_2) \cap [\kappa_2](E_1, E_2)$$

[7] If we let $\oplus$ be the element wise set union, define an event as a set of port assignments, and in the pair $([\kappa_\sqcap^{sync}], \oplus)$ let $\sqcap$ be true if and only if all common ports get the same value assigned, then this composition operator produces results similar to the composition operation in Reo [3].

[8] The lifting does not distribute across the union, however.

Proof.

$$\begin{aligned}
[\kappa_1](E_1,E_2)\cap[\kappa_2](E_1,E_2) =& \bigcup\{\mathcal{R} \mid \mathcal{R}\subseteq\Phi_{\kappa_1}(E_1,E_2)(\mathcal{R})\}\cap \\
& \bigcup\{\mathcal{R} \mid \mathcal{R}\subseteq\Phi_{\kappa_2}(E_1,E_2)(\mathcal{R})\} \\
=& \bigcup\{\mathcal{R} \mid \mathcal{R}\subseteq\Phi_{\kappa_1}(E_1,E_2)(\mathcal{R}) \textit{ and} \\
& \qquad \mathcal{R}\subseteq\Phi_{\kappa_2}(E_1,E_2)(\mathcal{R})\} \\
=& \bigcup\{\mathcal{R} \mid \mathcal{R}\subseteq\Phi_{\kappa_1}(E_1,E_2)(\mathcal{R})\cap\Phi_{\kappa_2}(E_1,E_2)(\mathcal{R})\} \\
=& \bigcup\{\mathcal{R} \mid \mathcal{R}\subseteq\Phi_{\kappa_1\cap\kappa_2}(E_1,E_2)(\mathcal{R})\} \\
=& [\kappa_1\cap\kappa_2](E_1,E_2)\}
\end{aligned}$$

since

$$\begin{aligned}
\Phi_{\kappa_1}(E_1,E_2)(\mathcal{R})\cap\Phi_{\kappa_2}(E_1,E_2)(\mathcal{R}) &= \{(\tau_1,\tau_2) \mid (\tau_1(0),\tau_2(0))\in\kappa_1(E_1,E_2)\wedge \\
&\qquad (\tau_1(0),\tau_2(0))\in\kappa_2(E_1,E_2)\wedge \\
&\qquad (\tau_1,\tau_2)'\in\mathcal{R})\} \\
&= \{(\tau_1,\tau_2) \mid (\tau_1(0),\tau_2(0))\in\kappa_1(E_1,E_2)\cap\kappa_2(E_1,E_2)\wedge \\
&\qquad (\tau_1,\tau_2)'\in\mathcal{R})\} \\
&= \Phi_{\kappa_1\cap\kappa_2}(E_1,\,E_2)(\mathcal{R})
\end{aligned}$$

□

Similarly, we give a mechanism to lift a composition function on observables to a composition function on TESs. Such lifting operation interleaves observations with different time stamps, and composes observations that occur at the same time.

Definition 15 (Lifting - composition function)**.** *Let* $+ : \mathcal{P}(\mathbb{E})\times\mathcal{P}(\mathbb{E})\to\mathcal{P}(\mathbb{E})$ *be a composition function on observables. The* lifting *of* $+$ *to TESs is* $[+] : TES(\mathbb{E})\times TES(\mathbb{E})\to TES(\mathbb{E})$ *such that, for* $\sigma_i\in TES(\mathbb{E})$ *where* $\sigma_i(0)=(O_i,t_i)$ *with* $i\in\{1,2\}$*:*

$$\sigma_1[+]\sigma_2 = \begin{cases} \langle\sigma_1(0)\rangle\cdot(\sigma_1'[+]\sigma_2) & \textit{if } t_1<t_2 \\ \langle\sigma_2(0)\rangle\cdot(\sigma_1[+]\sigma_2') & \textit{if } t_2<t_1 \\ \langle(O_1+O_2,t_1)\rangle\cdot(\sigma_1'[+]\sigma_2') & \textit{otherwise} \end{cases}$$

Definition 15 composes observations only if their time stamp is the same. Alternative definitions might consider time intervals instead of exact times.

Remark 2. *The last clause of Definition 15 considers the case where two observations occur at the same time. Recall that the time of an observation, as introduced earlier, is an abstraction that requires every event of the observation to occur after the events of the previous observation, and before the events of the next observation. Moreover, the time of two related observations, during composition, may be constrained by the interaction signature of the composition. For instance, the synchronous composability relation in Example 4 requires related observations to occur at the same time. Given those two facts, the likelihood that two observations have the same time is non zero.*

Lemma 9. *Let κ_1 and κ_2 be two composability relations and $\times_{([\kappa_1 \cap \kappa_2],\oplus)}$ be a product on components. Then,*

$$C_1 \times_{([\kappa_1 \cap \kappa_2],\oplus)} C_2 = C_1 \times_{([\kappa_1] \cap [\kappa_2],\oplus)} C_2 = (C_1 \times_{([\kappa_1],\oplus)} C_2) \cap (C_1 \times_{([\kappa_2],\oplus)} C_2)$$

Proof. Let $C_1 \times_{[\kappa_1 \cap \kappa_2],\oplus} C_2 = (E, L)$ and $(C_1 \times_{([\kappa_1],\oplus)} C_2) \cap (C_1 \times_{([\kappa_2],\oplus)} C_2) = (E', L')$. We have $E = E_1 \cup E_2 = E'$. We show $L = L'$.

$$\begin{aligned} L &= \{\sigma_1 \oplus \sigma_2 \mid \sigma_1 \in L_1, \sigma_2 \in L_2, (\sigma_1, \sigma_2) \in [\kappa_1 \cap \kappa_2](E_1, E_2)\} \\ &= \{\sigma_1 \oplus \sigma_2 \mid \sigma_1 \in L_1, \sigma_2 \in L_2, (\sigma_1, \sigma_2) \in [\kappa_1](E_1, E_2) \cap [\kappa_2](E_1, E_2)\} \\ &= L' \end{aligned}$$

□

Example 14. *Composability relations as defined in Definition 13 and Definition 14 can be combined to form new relations, and therefore new products. The behavior of component $C_1 \times_{([\kappa_\sqcap^{sync} \cap \kappa_\sqcap^{excl}]),\oplus)} C_2$ contains all TESs that are in the behavior of both $C_1 \times_{([\kappa_\sqcap^{sync}],\oplus)} C_2$ and $C_1 \times_{([\kappa_\sqcap^{excl}],\oplus)} C_2$, which excludes observations containing an occurrence of at least one event related by $\sqcap$.* ■

Co-inductive constructions of interaction signatures make proving algebraic results of Lemma 3 easier. Lemma 10 gives sufficient conditions to lift, by co-induction, properties of the underlying relation and composition function to meet the conditions of Lemma 3.

Lemma 10. *Let $\Sigma = ([\kappa], [+])$ be an interaction signature with $+$ be a composition function on observables and let κ be a composability relation on observations. Then,*

- $\times_\Sigma$ *is* commutative *if κ is symmetric and $+$ is commutative;*

- $\times_\Sigma$ *is* associative *if* $+$ *is associative and, for all* E_1, E_2, E_3 *and for all triple* $O_i \in \mathcal{P}(E_i)$ *with* $i \in \{1,2,3\}$ *and for all* $t \in \mathbb{R}_+$*:*

$$((O_1,t),(O_2,t)) \in \kappa(E_1,E_2) \wedge ((O_1+O_2,t),(O_3,t)) \in \kappa(E_1 \cup E_2, E_3)$$

if and only if

$$((O_2,t),(O_3,t)) \in \kappa(E_2,E_3) \wedge ((O_1,t),(O_2+O_3,t)) \in \kappa(E_1, E_2 \cup E_3)$$

with κ *such that* $((O_1,t_1),(O_2,t_2)) \in \kappa(E_1,E_2)$ *implies*

1. $((O_1,t_1),(\emptyset,t_2)) \in \kappa(E_1,E_2)$ *if* $t_1 < t_2$*; and*
2. $((\emptyset,t_1),(O_2,t_2)) \in \kappa(E_1,E_2)$ *if* $t_2 < t_1$*;*

and with $O + \emptyset = \emptyset + O = O$ *for all* $O \subseteq \mathcal{P}(\mathbb{E})$*.*

- $\times_\Sigma$ *is* idempotent *if* $+$ *is idempotent and, for all* $E \subseteq \mathbb{E}$ *we have* $((O_1,t_1),(O_2,t_2)) \in \kappa(E,E) \implies (O_1,t_1) = (O_2,t_2)$*.*

Proof. Commutativity. From Lemma 6, if κ is symmetric, then its lifting $[\kappa]$ is also symmetric. Therefore, it is sufficient for κ to be symmetric and for $+$ to be commutative in order for $[\kappa]$ to be symmetric and $[+]$ to be commutative, and therefore $\times_{([\kappa],[+])}$ to be commutative.

Associativity. We recall that, for a TES σ, the domain of σ is the collection of all time stamps of observations, i.e., $dom(\sigma) = \{t \mid \exists i \in \mathbb{N}.\sigma(i) = (O,t)\}$. We define a domain equalizer function $\equiv$ that, given a tuple $(\sigma_1,...,\sigma_n)$ returns a new tuple $(\sigma_1,...,\sigma_n)_\equiv = (\tau_1,...,\tau_n)$ such that the domains of TESs τ_i are the same, i.e., $\exists c.dom(\tau_i) = c$; and τ_i differs with σ_i only by empty observations, i.e., for all $t \in \mathbb{R}_+$, $\sigma_i(t) = \tau_i(t)$. Thus, the operation $\equiv$ adds silent observations to all TESs of the tuple such that the resulting domain for all τ_i is equal.

Let us first introduce a property induced by the assumptions on the composability relation κ. The fact that, forall $O_1 \subseteq E_1$ and $O_2 \subseteq E_2$, $((O_1,t_1),(O_2,t_2)) \in \kappa(E_1,E_2) \wedge t_1 < t_2 \iff ((O_1,t_1),(\emptyset,t_1)) \in \kappa(E_1,E_2)$ implies that a pair of TESs (σ_1,σ_2) is related by $[\kappa]$ if and only if their extension with silent observations on equal domains, i.e., $(\sigma_1,\sigma_2)_\equiv$, is also related by $[\kappa]$. We show by co-induction such property.

For $R \subseteq TES(E_1) \times TES(E_2)$, let

$$\begin{aligned}\Phi^{\equiv}_{k}(R) = \{(\sigma_1,\sigma_2) \mid \sigma_1(0) = (O_1,t_1) \wedge \sigma_2(0) = (O_2,t_2) \wedge \\ t_1 < t_2 \implies (\sigma_1(0),(\emptyset,t_1)) \in \kappa(E_1,E_2) \wedge \\ t_2 < t_1 \implies ((\emptyset,t_2),\sigma_2(0)) \in \kappa(E_1,E_2) \wedge \\ t_1 = t_2 \implies (\sigma_1(0),\sigma_2(0)) \in \kappa(E_1,E_2) \wedge \\ (\sigma_1,\sigma_2)' \in R\}\end{aligned}$$

The function $\Phi^{\equiv}_{\kappa}$ is a monotonous function applied on a complete lattice. By Knaster Tarksi theorem, $\Phi^{\equiv}_{\kappa}$ has a greatest fixed point that we call $[\kappa_{\equiv}]$. Given the assumption on κ, we can show that $[\kappa] \subseteq \Phi^{\equiv}_{\kappa}([\kappa])$ and $[\kappa_{\equiv}] \subseteq \Phi_{\kappa}([\kappa_{\equiv}])$, which implies that $[\kappa] = [\kappa_{\equiv}]$. It follows that $(\sigma_1,\sigma_2) \in [\kappa] \iff (\sigma_1,\sigma_2)_{\equiv} \in [\kappa]$.

A sufficient condition for the product $\times_{([\kappa],[+])}$ to be associative is that $+$ is associative and for every $\sigma_i \in TES(E_i)$ for $i \in \{1,2,3\}$:

$$\begin{aligned}P_1 :=&(\sigma_1,\sigma_2) \in [\kappa](E_1,E_2) \wedge (\sigma_1[+]\sigma_2,\sigma_3) \in [\kappa](E_1 \cup E_2,E_3) \iff \\ &(\sigma_2,\sigma_3) \in [\kappa](E_2,E_3) \wedge (\sigma_1,\sigma_2[+]\sigma_3) \in [\kappa](E_1,E_2 \cup E_3)\end{aligned}$$

which is equivalent to

$$\begin{aligned}&(\sigma_1,\sigma_2) \in [\kappa_{\equiv}](E_1,E_2) \wedge (\sigma_1[+]\sigma_2,\sigma_3) \in [\kappa_{\equiv}](E_1 \cup E_2,E_3) \iff \\ &(\sigma_2,\sigma_3) \in [\kappa_{\equiv}](E_2,E_3) \wedge (\sigma_1,\sigma_2[+]\sigma_3) \in [\kappa_{\equiv}](E_1,E_2 \cup E_3)\end{aligned}$$

and with similar arguments as before, is equivalent to

$$\begin{aligned}&(\tau_1,\tau_2) \in [\kappa_{\equiv}](E_1,E_2) \wedge (\tau_1[+]\tau_2,\tau_3) \in [\kappa_{\equiv}](E_1 \cup E_2,E_3) \iff \\ &(\tau_2,\tau_3) \in [\kappa_{\equiv}](E_2,E_3) \wedge (\tau_1,\tau_2[+]\tau_3) \in [\kappa_{\equiv}](E_1,E_2 \cup E_3)\end{aligned}$$

with $(\tau_1,\tau_2,\tau_3) = (\sigma_1,\sigma_2,\sigma_3)_{\equiv}$.

As we can assume that all TESs in a triple satisfying P_1 have the same domain, it is sufficient to show that, for all $O_i \subseteq E_i$ with $i \in \{1,2,3\}$ and all $t \in \mathbb{R}_+$:

$$((O_1,t),(O_2,t)) \in \kappa(E_1,E_2) \wedge ((O_1+O_2,t),(O_3,t)) \in \kappa(E_1 \cup E_2,E_3)$$

if and only if

$$((O_2,t),(O_3,t)) \in \kappa(E_2,E_3) \wedge ((O_1,t),(O_2+O_3,t)) \in \kappa(E_1,E_2 \cup E_3)$$

which is assumed by κ. Thus, given the properties of κ, P_1 holds.

Finally, we prove that if $+$ is associative, then $[+]$ is associative. Let $\sigma_i \in L_i$ and we write $\sigma_i(0) = (O_i, t_i)$ for $i \in \{1,2,3\}$, then:

$$\sigma_1[+](\sigma_2[+]\sigma_3) = \begin{cases} \langle (O_1,t_1)\rangle \cdot (\sigma_1'[+](\sigma_2[+]\sigma_3) & \text{if } t_1 < t_2, t_3 \\ \langle (O_2,t_2)\rangle \cdot (\sigma_1[+](\sigma_2'[+]\sigma_3) & \text{if } t_2 < t_1, t_3 \\ \langle (O_3,t_3)\rangle \cdot (\sigma_1[+](\sigma_2[+]\sigma_3') & \text{if } t_3 < t_2, t_1 \\ \langle (O_1+O_2,t_1)\rangle \cdot (\sigma_1'[+](\sigma_2'[+]\sigma_3) & \text{if } t_1 = t_2 < t_3 \\ \langle (O_2+O_3,t_2)\rangle \cdot (\sigma_1[+](\sigma_2'[+]\sigma_3') & \text{if } t_2 = t_3 < t_1 \\ \langle (O_1+O_3,t_1)\rangle \cdot (\sigma_1'[+](\sigma_2[+]\sigma_3') & \text{if } t_1 = t_3 < t_2 \\ \langle (O_1+(O_2+O_3),t_1)\rangle \cdot (\sigma_1'[+](\sigma_2'[+]\sigma_3') & \text{if } t_1 = t_3 = t_2 \end{cases}$$

The only case that differs from $(\sigma_1[+]\sigma_2)[+]\sigma_3$ is when $t_1 = t_3 = t_2$, which gives $((O_1+O_2)+O_3,t_1)$. Thus, if $((O_1+O_2)+O_3,t_1) = (O_1+(O_2+O_3),t_1)$ for every $O_i \in \mathcal{P}(E_i)$ with $i \in \{1,2,3\}$, then $\sigma_1[+](\sigma_2[+]\sigma_3) = \sigma_1[+](\sigma_2[+]\sigma_3)$ for every $\sigma_i \in L_i$ with $i \in \{1,2,3\}$.

Idempotency. If $+$ is idempotent, then the lifting $[+]$ is also idempotent. We consider $+$ to be idempotent. We show that, for all $E \subseteq \mathbb{E}$ and $o_1, o_2 \in \mathcal{P}(E) \times \mathbb{R}_+$ we have $(o_1,o_2) \in \kappa(E,E) \implies o_1 = o_2$, then for all $\sigma, \tau \in TES(E)$, $(\sigma,\tau) \in [\kappa](E,E) \implies \sigma = \tau$, which is a sufficient condition for $\times_{([\kappa],[+])}$ to be idempotent.

By definition $[\kappa](E,E)$ is the greatest fixed point of the function:

$$\begin{aligned} \Phi_\kappa(E,E)(\mathcal{R}) &= \{(\tau_1,\tau_2) \mid \quad (\tau_1(0),\tau_2(0)) \in \kappa(E,E) \wedge (\tau_1,\tau_2)' \in \mathcal{R})\} \\ &\subseteq \{(\tau_1,\tau_2) \mid \quad \tau_1(0) = \tau_2(0) \wedge (\tau_1',\tau_2') \in \mathcal{R}\} \end{aligned}$$

Therefore, we conclude that $[\kappa](E,E) \subseteq \{(\sigma,\sigma) \mid \sigma \in TES(E)\}$. □

The conditions exposed in Lemma 10 are applicable for the case of the join product, as shown in Theorem 1.

Theorem 1. *The product $\bowtie$ of Example 7 is commutative, associative, and idempotent.*

Proof. Commutativity and idempotency of $\bowtie$ are following from κ^{sync} being symmetric and satisfying the condition for idempotency.

Assume that $((O_1,t),(O_2,t)) \in \kappa^{sync}(E_1,E_2) \wedge ((O_1 \cup O_2,t),(O_3,t)) \in \kappa^{sync}(E_1 \cup E_2,E_3)$ holds, then $O_1 \cap E_2 = O_2 \cap E_1 \wedge (O_1 \cup O_2) \cap E_3 = O_3 \cap (E_1 \cup E_2)$ is true by definition of κ^{sync}.

We first observe that $(O_1 \cup O_2) \cap E_3 \cap E_2 = O_3 \cap (E_1 \cup E_2) \cap E_2$ implies that $O_2 \cap E_3 = O_3 \cap E_2$. Then, $(O_1 \cup O_2) \cap E_3 \cap E_1 = O_3 \cap (E_1 \cup E_2) \cap E_1$ implies that $O_1 \cap E_3 = O_3 \cap E_1$, using $O_1 \cap E_2 = O_2 \cap E_1$ we conclude that $O_1 \cap (E_1 \cup E_3) = (O_2 \cup O_3) \cap E_1$.

Thus, we showed that

$$((O_1,t),(O_2,t)) \in \kappa^{sync}(E_1,E_2) \wedge ((O_1 \cup O_2,t),(O_3,t)) \in \kappa^{sync}(E_1 \cup E_2,E_3)$$

if and only if

$$((O_2,t),(O_3,t)) \in \kappa^{sync}(E_2,E_3) \wedge ((O_1,t),(O_2 \cup O_3,t)) \in \kappa^{sync}(E_1,E_2 \cup E_3)$$

for all $O_i \subseteq E_i$ and $t \in \mathbb{R}_+$. Finally, by definition, κ_{sync} is such that, for all $O_1 \subseteq E_1$ and $O_2 \subseteq E_2$:

1. $((O_1,t_1),(O_2,t_2)) \in \kappa_{sync}(E_1,E_2)$ and $t_1 < t_2$ if and only if $((O_1,t_1),(\emptyset,t_1)) \in \kappa_{sync}(E_1,E_2)$; and

2. $((O_1,t_1),(O_2,t_2)) \in \kappa_{sync}(E_1,E_2)$ and $t_2 < t_1$ if and only if $((\emptyset,t_2),(O_2,t_2)) \in \kappa_{sync}(E_1,E_2)$.

Given that $\cup$ is associative and $O \cup \emptyset = O$ for all O, we conclude that $\bowtie$ is associative. □

We give in Lemma 11 some conditions for two products to distribute, and in Lemma 12 some conditions to extend the underlying relation on observables for a synchronous composability relation.

Lemma 11. *Let C_1, C_2, and C_3 be three components, and let κ_1 and κ_2 be two composability relations on observables such that for all $\sigma_1,\sigma_2,\sigma_3 \in L_1 \times L_2 \times L_3$:*

- *$(\sigma_1,\sigma_2[\cup]\sigma_3) \in [\kappa_1]$ if and only if $(\sigma_1,\sigma_2) \in [\kappa_1]$ and $(\sigma_1,\sigma_3) \in [\kappa_1]$, and*

- *for all $\tau_1 \in L_1$, $(\tau_1[\cup]\sigma_2,\sigma_1[\cup]\sigma_3) \in [\kappa_2]$ if and only if $(\sigma_2,\sigma_3) \in [\kappa_2]$ and $\sigma_1 = \tau_1$.*

Then,

$$C_1 \times_{[\kappa_1]} (C_2 \times_{[\kappa_2]} C_3) = (C_1 \times_{[\kappa_1]} C_2) \times_{[\kappa_2]} (C_1 \times_{[\kappa_1]} C_3)$$

Proof. Let L be the behavior of component $(C_1 \times_{[\kappa_1]} C_2) \times_{[\kappa_2]} (C_1 \times_{[\kappa_1]} C_3)$, L' be the behavior of $C_1 \times_{[\kappa_1]} (C_2 \times_{[\kappa_2]} C_3)$, L_{12} be the behavior of $(C_1 \times_{[\kappa_1]} C_2)$ and L_{13} be the behavior of $(C_1 \times_{[\kappa_1]} C_3)$. Then,

$$\begin{aligned}
L =&\{\sigma_1[\cup](\sigma_2[\cup]\sigma_3) \mid \sigma_1 \in L_1, \sigma_2 \in L_2, \sigma_3 \in L_3, \\
&\qquad (\sigma_1, \sigma_2[\cup]\sigma_3) \in [\kappa_1],\ (\sigma_2, \sigma_3) \in [\kappa_2]\} \\
=&\{\sigma_1[\cup](\sigma_2[\cup]\sigma_3) \mid \sigma_1 \in L_1, \sigma_2 \in L_2, \sigma_3 \in L_3, \\
&\qquad (\sigma_1, \sigma_2) \in [\kappa_1], (\sigma_1, \sigma_3) \in [\kappa_1], (\sigma_2, \sigma_3) \in [\kappa_2]\} \\
=&\{\sigma[\cup]\tau \mid \sigma \in L_{12}, \tau \in L_{13}, (\sigma, \tau) \in [\kappa_2]\} \\
=&L'
\end{aligned}$$

□

Lemma 12. *Let $C_1 = (E_1, L_1)$ and $C_2 = (E_2, L_2)$ be two components. Let $\kappa_\sqcap^{sync}$ be a composability relation on observables with $\sqcap \subseteq \mathcal{P}(E_1) \times \mathcal{P}(E_2)$. Then, for any $\sqcap'$ with $\sqcap' \cap (\mathcal{P}(E_1) \times \mathcal{P}(E_2)) = \emptyset$, then:*

$$C_1 \times_{[\kappa_\sqcap^{sync}]} C_2 = C_1 \times_{[\kappa_{\sqcap\cup\sqcap'}^{sync}]} C_2$$

Proof. For any pair of composable observations $((O_1, t_1), (O_2, t_2)) \in \kappa_\sqcap^{sync}$, we have that $((O_1, t_1), (O_2, t_2)) \in \kappa_{\sqcap\cup\sqcap'}^{sync}$ since $(O_1, O_2) \in \sqcap$ implies that $(O_1, O_2) \in \sqcap \cup \sqcap'$. Conversely, if $(O_1, O_2) \in \sqcap\cup\sqcap'$ and $\sqcap' \cap \mathcal{P}(E_1) \times \mathcal{P}(E_2) = \emptyset$, then $(O_1, O_2) \in \sqcap$. Thus, for any (σ_1, σ_2), $(\sigma_1, \sigma_2) \in [\kappa_\sqcap^{sync}]$ if and only if $(\sigma_1, \sigma_2) \in [\kappa_{\sqcap\cup\sqcap'}^{sync}]$. □

2.1.5 Properties of TESs

We distinguish two kinds of properties of TESs: properties that we call *trace properties*, and properties on sets of TESs that we call *behavior properties*, which correspond to hyper-properties in [28]. The generality of our model permits to interchangeably construct a component from a property and extract a property from a component. As illustrated in Example 16, when composed with a set of interacting components, a component property constrains the components to only expose desired behavior (i.e., behavior in the property). In Section 2.1.6, we provide more intuition for the practical relevance of these properties.

Definition 16. *A trace property P is a subset $P \subseteq TES(E)$ for some set of events E. A component $C = (E, L)$ satisfies a property P, if $L \subseteq P$, which we denote as $C \models P$.*

Example 15. *We distinguish the usual* safety *and* liveness *properties [2, 28], and recall that every trace property can be written as the intersection of a safety and a liveness property. Let X be an arbitrary set, and P be a subset of $\mathbb{N} \to X$. Intuitively, P is safe if every* bad *stream not in P has a finite prefix every completion of which is bad, hence not in P. A property P is a liveness property if every finite sequence in X^* can be completed to yield an infinite sequence in P, where X^* is the set of all finite sequences of elements in X. For instance, the property of terminating behavior for a component with interface E is a liveness property, defined as:*

$$P_{finite}(E) = \{\sigma \in TES(E) \mid \exists n \in \mathbb{N}. \forall i > n.\ \mathsf{pr}_1(\sigma)(i) = \emptyset\}$$

$P_{finite}(E)$ says that, for every finite prefix of any stream in $TES(E)$, there exists a completion of that prefix with an infinite sequence of silent observations $\emptyset$ in $P_{finite}(E)$. ■

Definition 17. *A trace property is similar to a component, since it describes a set of TESs, except that it is* a priori *not restricted to any interface* [9]*. A trace property P can then be turned into a component, by constructing the smallest interface E_P such that, for all $\sigma \in P$, and $i \in \mathbb{N}$, $\mathsf{pr}_1(\sigma)(i) \subseteq E_P$. The component $C_P = (E_P, P)$ is then the componentized-version of property P.* ■

Lemma 13. *Given a property P over E, its componentized-version C_P (see Definition 17) and a component $C = (E, L)$, then $C \models P$ if and only if $C \cap C_P = C$.*

Proof. We recall the definition of the intersection in Definition 12. For any two components $C_1 = (E_1, L_1)$ and $C_2 = (E_2, L_2)$, the intersection $C_1 \cap C_2$ is the component $C_1 \times_{([\kappa^{sync}],[\cap])} C_2 = (E_1 \cup E_2, L)$. Given that $\cap$ satisfies the condition for using Lemma 10, the product $\cap$ is idempotent. Let $C \cap C_P = (E, L')$. If $(\sigma, \tau) \in L'$ then $\sigma = \tau$. Thus, $L' \subseteq L \cap L_P$.

Alternatively, let $\sigma \in L \cap L_P$. We observe that at any point $n \in \mathbb{N}$, we have $(\sigma(n), \sigma(n)) \in \kappa^{sync}(E, E)$. Therefore, $(\sigma, \sigma) \in [\kappa^{sync}_{\sqcap}]$.

We conclude that $L \cap L_P = L'$. □

[9] In our formalism, a property is a set of TESs $L \subseteq TES(E)$ for some $E \subseteq \mathbb{E}$. Two properties P and L are equal if they contain identical TESs, and equality is not subject to the interface over which properties are defined.

Example 16. *We use the term* coordination property *to refer to a property used in order to coordinate behaviors. Given a set of n components $C_i = (E_i, L_i)$, $i \in \{1, ..., n\}$, a coordination property Coord for the composed components is a property over events $E = E_1 \cup ... \cup E_n$, i.e., $Coord \subseteq TES(E)$.*

Consider the synchronous interaction, as introduced in Example 4, of the n components and let $C = C_1 \bowtie C_2 \bowtie ... \bowtie C_n$ be their synchronous product. Typically, a coordination property will not necessarily be satisfied by the composite component C, but some of the behavior of C is contained in the coordination property. The coordination problem is to find (e.g., synthesize) an orchestrator component $Orch = (E_O, L_O)$ such that $C \bowtie Orch \models Coord$. The orchestrator restricts the component C to exhibit only the subset of its behavior that satisfies the coordination property. In other words, in their composition, Orch coordinates C to satisfy Coord. As introduced in Definition 17, since Coord ranges over the same set E that is the interface of component $C_1 \bowtie C_2 \bowtie ... \bowtie C_n$, a coordination property can be turned into an orchestrator by building its corresponding component. The coordination problem can be made even more general by changing the composability relations or the composition functions used in the construction of C. ■

Trace properties are not sufficient to fully capture the scope of interesting properties of components of cyber-physical systems. Some of their limitations are highlighted in Section 2.1.6. To address this issue, we introduce *behavior properties*, which are strictly more expressive than trace properties, and give two illustrative examples.

Definition 18. *A behavior property ϕ over a set of events E is a hyper-property $\phi \subseteq \mathcal{P}(TES(E))$. A component $C = (E, L)$ satisfies a hyper-property ϕ if $L \in \phi$, which we denote as $C \models\!\models \phi$.*

Example 17. *A component $C = (E, L)$ can be oblivious to time. Any sequence of time-stamps for an acceptable sequence of observables is acceptable in the behavior of such a component. This "obliviousness to time" property is not a trace property, but a hyper-property, defined as:*

$$\phi_{shift}(E) := \{Q \subseteq TES(E) \mid \forall \sigma \in Q. \forall t \in OS(\mathbb{R}_+). \exists \tau \in Q. \\ \mathsf{pr}_1(\sigma) = \mathsf{pr}_1(\tau) \wedge \mathsf{pr}_2(\tau) = t\}$$

Intuitively, if $C \models\!\models \phi_{shift}(E)$, then C is independent of time. ■

Example 18. *We use $\phi_{insert}(X, E)$ to denote the hyper-property that allows for arbitrary insertion of observations in $X \subseteq \mathcal{P}(E)$ into every TES at any point in time,*

i.e., the set defined as:

$$\{Q \subseteq TES(E) \mid \forall \sigma \in Q.\forall i \in \mathbb{N}.\exists \tau \in Q.\exists x \in X. \\ (\forall j < i.\sigma(j) = \tau(j)) \wedge \\ (\exists t \in \mathbb{R}_+.\tau(i) = (x,t)) \wedge \\ (\forall j \geq i.\tau(j+1) = \sigma(j))\}$$

Intuitively, elements of $\phi_{insert}(X,E)$ are closed under insertion of an observation $x \in X$ at an arbitrary time. ■

2.1.6 Components of the running example

This section is inspired by the work on soft-agents [84, 50], and elaborates on the more intuitive version that we presented in Section 1.3. Interactive cyber-physical systems are represented as components, and behavioral properties of those systems are formulated as components, as well. Through these examples, we show how we use component-based descriptions to model a simple scenario of a robot roaming around in a field while taking energy from its battery. We structurally separate the battery, the robot, and the field as independent components, and we explicitly model their interaction in a specific composed system.

Example 19 (Roaming robots)**.** *We capture, as a component, sequences of observations emerging from discrete actions of a robot at a fixed time rate. For simplicity, we consider that the robot can perform actions of two types only: a move in a cardinal direction, and a read of its position sensor. A move action of robot i creates an event of the form $d(i,p)$ where d is the direction, and p is the power displayed by the robot. The read action of robot i generates an event of the form $read(i,(x;y))$ where $(x;y)$ is a coordinate location.*

Formally, we write $R(i,T,P) = (E_R(i,P), L_R(T))$ for the robot component with identifier i with $E_R(i,P)$ the set

$$\{S(i,p), W(i,p), N(i,p), E(i,p), read(i,(x;y)) \mid x,y \in [\![-20,20]\!], p \leq P\}$$

and $L_R(T) \subseteq TES(E_R(i,P))$ be such that all observations are time stamped with a multiple of the period $T \in \mathbb{R}_+$, i.e., for all $\sigma \in L_R(T)$, if $(O,t) \in \sigma$ then there exists $k \in \mathbb{N}$ such that $t = k \cdot T$. The component $R(i,T)$ therefore captures all robots whose directions are restricted to S(outh), W(est), N(orth), and E(ast), whose power

Table 2.1: Three prefixes of timed-event streams for $R(1,T,P)$, $R(2,T,P)$, and $R(3,T,P)$, where T and P are fixed, and each move action consumes the same power p.

t/T	$\sigma : R(1,T,P)$	$\tau : R(2,T,P)$	$\delta : R(3,T,P)$
1	$\{N(1,p)\}$	–	–
2	$\{W(1,p)\}$	–	–
3	$\{W(1,p)\}$	$\{N(2,p)\}$	$\{E(3,p)\}$
4	$\{S(1,p)\}$	$\{W(2,p)\}$	$\{E(3,p)\}$
...	...	...	–

is limited to P, and whose location values are integers in the interval $[\![-20,20]\!]$. The robot stops whenever $p = 0$.

In Table 2.1, we display the prefix of one TES from the behavior of three robot components. Note that each line corresponds to a time instant, for which each robot may or may not have observed events. The symbol '–' represents no observable, while otherwise we show the set of events observed. The time column is factorized by the period T, shared by all robots. Thus, at time $3 \cdot T$, robot $R(1,T,P)$ moves west, while robot $R(2,T,P)$ moves north, and robot $R(3,T,P)$ moves east, all with power p. We use $R(i)$ to denote the robot $R(i,T,P)$ with a fixed by arbitrary period T and upper power P. ■

In the robot component of Example 19, observations occur at a fixed frequency. For some physical components, however, observations may occur at any point in time. For instance, consider the field on which the robot moves. Each time a robot moves may induce a change in the field's state, and the field's state may be observable at any time and frequency. Internally, the field may record its state changes by a continuous function, while restricting the possibilities of the robots to move due to physical limitations. We describe the field on which the robot moves as a component, and we specify, in Example 22, how robot components interact with the field component.

Example 20 (Field)**.** *The field component captures, in its behavior, the dynamics of its state as a sequence of observations. The collection of objects on a field is given by the set I. The state of a field is a triple $(((x;y)_i)_{i\in I}, (\overrightarrow{v_i})_{i\in I}, t)$ that describes, at time $t \in \mathbb{R}_+$, the position $(x;y)_i$ and the velocity $\overrightarrow{v_i}$ of each object in I. We model each object in I by a square of dimension 1 by 1, and the coordinate $(x;y)_i$ represents the central position of the square. We use $\mu = (((x_0,y_0)_i)_{i\in I}, (\overrightarrow{v_{0i}})_{i\in I}, t_0)$ as an initial state for the field, which gives for each robot in $i \in I$ a position and an initial velocity. Note that static obstacles on the field can be modeled as objects $i \in I$ with position $(x;y)_i$ and zero velocity.*

Formally, the field component is the pair $F_\mu(I) = (E_F(I), L_F(I,\mu))$ *with*

$$E_F(I) = \{(x;y)_i, move(i,\overrightarrow{v}) \mid i \in I,\ x,y \in \mathbb{R},\ \overrightarrow{v} \in \mathbb{R}\times\mathbb{R}\}$$

where each event $move(i,\overrightarrow{v})$ *continuously moves object* i *with velocity* $\overrightarrow{v}$*, and event* $(x;y)_i$ *displays the location of object* i *on the field.*

The set $L_F(I,\mu) \subseteq TES(E_F(I))$ *captures all sequences of observations that consistently sample trajectories of each objects in* I*, according to the change of state of the field and the internal constraint. As a physical constraint, we impose that no two objects can overlap, i.e., for any disjoint* $i,j \in I$ *and for all time* $t \in \mathbb{R}_+$*, with* $(x;y)_i$ *and* $(u;v)_j$ *their respective positions, then* $[x-0.5,x+0.5]\cap[u-0.5,u+0.5] = [y-0.5,y+0.5]\cap[v-0.5,v+0.5] = \emptyset$*. Even though the mechanism for such a constraint is hidden in the field component, typically, the move of a robot is eventually limited by the physics of the field. We write* $F(I)$ *when* μ *is fixed but arbitrary.* ■

There is a fundamental difference between the robot component in Example 19 and the field component in Example 20. The robot component has an adequate underlying sampling frequency which prevents missing any event if observations are made by that frequency. However, the field has no such frequency for its observations, which means that there may be another intermediate observation occurring between any two observations. In [62], we capture, as a behavioral property, the property for a component to interleave observations between any two observation.

Example 21 (Protocol)**.** *As shown in Example 20, physics may impose some constraints that force robots to coordinate. A protocol is a component that, for instance, coordinates the synchronous movement of a pair of robots. For example, when two robots face each other on the field, the* $swap(i,j)$ *protocol moves robot* $R(i,P,T)$ *north, west, and then south, as it moves robot* $R(j,P,T)$ *east. Note that the protocol requires the completion of a sequence of moves to succeed. Another robot could be in the way, and therefore delay the last observables of the sequence. The swap component is defined by* $swap(i,j) = (E_P(i,j), L_P(i,j))$ *where* $E_P(i,j) = E_R(i,P) \cup E_R(j,P)$ *and* $L_P(i,j)$ *captures all sequences of observations where the two robots* i *and* j *swap positions.* ■

A useful interaction signature Σ is the one that synchronizes shared events between two components. We write $\Sigma_{sync} = (R_{sync}, \cup)$ for such interaction signature, and give its specification in Example 4. The synchronous interaction signature $\Sigma_{sync} = (R_{sync}, \cup)$ leads to the product $\times_{\Sigma_{sync}}$ that forces two components to observe shared events at the same time. We write $\bowtie$ for such product. As a result, $R(1,P,T) \bowtie$

Table 2.2: Three prefixes of timed-event streams for $R(1,T,P)$, $R(2,T,P)$, and $R(3,T,P)$, together with the prefix resulting from forming their synchronous product with the swap protocol. Initially, $\mu(1) = (0;2)$, $\mu(2) = (0;1)$, $\mu(3) = (0;0)$.

t/T	$\eta : R(1,P,T) \bowtie R(2,P,T) \bowtie R(3,P,T) \bowtie swap(2,3)$
1	$\{N(1,p)\}$
2	$\{W(1,p)\}$
3	$\{W(1,p), N(2,p)\}$
4	$\{S(1,p), W(2,p), E(3,p)\}$
5	$\{S(2,p)\}$
...	...

$R(2,P,T) \bowtie R(3,P,T) \bowtie swap(2,3)$ captures all sequences of moves for the robots constrained by the *swap* protocol, as shown by the elements of table 2.2.

Example 22 (Field-Robot signature)**.** *The interactions occurring between the field and the robot components impose simultaneity on some disjoint events. For instance, every observation of the robot containing the event $d(i,p) \in E_R(i,P)$ must occur at the same time as an observation of the field containing the event $move(i, \overrightarrow{v(d,p)}) \in E_F(I)$ with $\overrightarrow{v(d,p)}$ returning the velocity as a function of direction d and power p. Also, every observation containing the event $read(i, (\lfloor x \rfloor, \lfloor y \rfloor)) \in E_R(i,P)$ must occur at the same time as an event $(x;y)_i \in E_F(I)$ where $\lfloor z \rfloor$ gives the integer part of z.*

Formally, we capture such interaction in the interaction signature $\Sigma_{RF} = (R_{RF}, \cup)$, where R_{RF} is the smallest symmetric relation defined as for all $(\tau, \sigma) \in R_{RF}$, for all $t \in \mathbb{R}_+$, for all $i \in \mathbb{N}$,

$$read(i,(n,m)) \in \tau(t) \iff (\exists (x;y)_i \in \sigma(t) \wedge n = \lfloor x \rfloor \wedge m = \lfloor y \rfloor)$$

and $d(i,p) \in \tau(t) \iff move(i, \overrightarrow{v(d,p)}) \in \sigma(t)$ with $d \in \{N,W,E,S\}$.

As a result, the product $(R(1,T,P) \bowtie R(2,T,P) \bowtie R(3,T,P)) \times_{\Sigma_{RF}} F_\mu(I)$ captures all sequences of observations for the three robots constrained by the field component. ■

Remark 3. *The floor part $\lfloor \cdot \rfloor$ acts as an approximation of the robot sensor on the field's position value. A different interaction signature may, for instance, introduce some errors in the reading.*

The interaction signature may also impose that $d(i,p)$ relates to the speed $(0, 1/T)$, $(0, -1/T)$, $(-1/T, 0)$, and $(1/T, 0)$ when $d = N$, $d = S$, $d = W$, and $d = E$, respectively. Then, for a time interval T, the power p moves the robot by one unit on the field.

Remark 4. *In practice, it is unlikely that two observations happen at exactly at the same time. However, in our framework, the time of an observation is an abstraction that requires every event of the observation to occur after the events of the previous observation, and before the events of the next observation.*

Example 23 (Battery)**.** *A battery component with capacity C in mAH is a pair $B = (E_B(C), L_B(C))$ with events $read(l) \in E_B(C)$ for $0\% \leq l \leq 100\%$, $charge(\mu) \in E_B(C)$, and $discharge(\mu) \in E_B(C)$ with μ a (dis)charging coefficient in % per seconds. The battery displays its capacity with the event capacity(C). The behavior L_B is a set of sequences $\sigma \in L_B$ such that there exists a piecewise linear function $f : \mathbb{R}_+ \to \mathcal{P}(E_B)$ with, for $\sigma(i) = (O_i, t_i)$,*

- *for $\sigma(0) = (O_0, t_0)$, $f([0; t_0]) = 100\%$, i.e., the battery is initially fully charged;*
- *if $O_i = \{read(l)\}$, then $f(t_i) = l$ and the derivation $f'_{[t_{i-1}, t_{i+1}]}$ of f is constant in $[t_{i-1}, t_{i+1}]$, i.e., the observation does not change the slope of f at time t_i;*
- *if $O_i = \{discharge(\mu)\}$, then $f_{[t_i, t_{i+1}]}(t) = \max(f(t_i) - (t - t_i)\mu, 0)$;*
- *if $O_i = \{charge(\mu)\}$, then $f_{[t_i, t_{i+1}]}(t) = \min(f(t_i) + (t - t_i)\mu, 100)$;*

where $f_{[t_1;t_2]}$ is the restriction of function f on the interval $[t_1; t_2]$. There is a priori *no restrictions on the time interval between two observations, as long as the sequence of timestamps is increasing and non-Zeno. Finally, we use B_i for a battery whose events are identified by the natural number i. Then, for $B = (E_B(C), L_B(C))$, we have $B_i = (E_{B_i}(C), L_{B_i}(C))$ with $e \in E_B(C)$ if and only if the corresponding identified event $e_i \in E_{B_i}(C)$, e.g. $read_i(l) \in E_{B_i}(C)$ for all $read(l) \in E_B(C)$. The set of TESs $L_{B_i}(C)$ is obtained by replacing in every TESs in $L_B(C)$ the corresponding identified event in $E_{B_i}(C)$.* □

Example 24. *We define $\Sigma_{RB} = ([\kappa_{RB}], \cup)$ where $\cup$ unions two TESs as defined in the preliminaries, and $[\kappa_{RB}]$ specifies co-inductively (see [62] for details of the construction), from a relation on observations κ_{RB}, how event occurrences relate in the robot and the battery components of capacity C. More specifically, κ_{RB} is the smallest symmetric relation over observations such that $((O_1, t_1), (O_2, t_2)) \in \kappa_{RB}$ implies that $t_1 = t_2$ and*

- *the discharge event in the battery coincides with a move of the robot, i.e., $d(i, p) \in O_1$ if and only if $discharge(\mu) \in O_2$. Moreover, the interaction signature imposes a relation between the discharge coefficient μ and the required power p, i.e., $\mu = p/C$;*

- *the read value of the robot sensor coincides with a value from the battery component, i.e., read$(i,l) \in O_1$ if and only if read$(l) \in O_2$;*
- *the robot reads the capacity value that corresponds to the battery capacity, i.e., getCapacity(i,c) $\in O_1$ if and only if capacity(c) $\in O_2$.*

The product $B \times_{\Sigma_{RB}} R(i,P,T)$ of a robot and a battery component, under the interaction signature Σ_{RB}, restricts the behavior of the battery to match the periodic behavior of the robot, and restricts the behavior of the robot to match the sensor values delivered by the battery.
As a result, the behavior of the product component $B \times_{\Sigma_{RB}} R(i,P,T)$ contains all observations that the robot performs in interaction with its battery. Note that trace properties, such as all energy sensor values observed by the robot are within a safety interval*, does not necessarily entail safety of the system: some unobserved energy values may fall outside of the safety interval. Moreover, the frequency by which the robot samples may* reveal *some new observations, and such robot can safely sample at period T if, for any period $T' \leq T$, the product $B \times_{\Sigma_{RB}} R(i,P,T')$ satisfies the safety property.*
In case of a battery B_j with identifier j and a robot $R(i)$ with identifier i, we use $\Sigma_{R_iB_j}$ for the interactio signature that synchronizes, as described above, occurrences of events of the battery B_j with occurrences of events of the robot $R(i)$. □

Behavioral properties of components

Consider the system

$$S(n,T_1,...,T_n) = \bowtie_{i\in\{1,...,n\}} (R(i,T_i) \times_{\Sigma_{R_iB_i}} B_i) \times_{\Sigma_{RF}} F(\{1,...,n\}) \tag{2.1}$$

made of n robots $R(i,T_i)$, each interacting with a private battery B_i under the interaction signatures $\Sigma_{R_iB_i}$, and in product with a field F under the interaction signature Σ_{RF}. We use $\bowtie$ for the product with the free interaction signature (i.e., every pair of TESs is composable), and the notation $\bowtie_{i\in\{1,...,n\}} \{C_i\}$ for $C_1 \bowtie ... \bowtie C_n$ as $\bowtie$ is commutative and associative.

We fix $n = 2$ in Equation (2.1) and the same period T for the two robots. We write E for the set of events of the composite system $S(2,T)$. We formulate the scenarios described in Section 1.3 in terms of a satisfaction problem involving a safety property on TESs and a behavioral property on the composite system. We first consider two

safety properties:

$$P_{energy} = \{\sigma \in TES(E) \mid \forall i \in \mathbb{N}.\{read_1(0), read_2(0)\} \not\subseteq \mathsf{pr}_1(\sigma)(i)\}$$

which models that the two batteries don't display simultaneously that their level is empty; and $P_{no-overlap}$ which is the set

$$\{\sigma \in TES(E) \mid \forall i \in \mathbb{N}.\forall (x,y) \in [0,20]^2, \{(x,y)_1, (x,y)_2\} \not\subseteq \mathsf{pr}_1(\sigma)(i)\}$$

that captures all behaviors where the two robots are never observed together at the same location.

Observe that, while both P_{energy} and $P_{no-overlap}$ specify some safety properties, they are not sufficient to ensure the safety of the system. We illustrate some scenarios with the property P_{energy}. If a component never reads its battery level, then the property P_{energy} is trivially satisfied, although effectively the battery may run out of energy. Also, if a component reads its battery level periodically, each of its readings may return an observation agreeing with the property. However, in between two read events, the battery may run out of energy (and somehow recharge). To circumvent those unsafe scenarios, we add an additional behavioral property.

Let $X_{read} = \{read(l_1)_1, read(l_2) \mid 0 \leq l_1 \leq C_1,\ 0 \leq l_2 \leq C_2\}$ be the set of reading events for battery components B_1 and B_2, with capacities C_1 and C_2 respectively. The property $\phi_{insert}(X_{read}, E)$, as detailed in Example 18, defines a class of component behaviors that are closed under insertion of *read* events for the battery component. Therefore, the system $S(2,T)$ is energy safe if $S(2,T) \models P_{energy}$ and its behavior is closed under insertion of battery read events, i.e., $S(2,T) \Vdash \phi_{insert}(X_{read}, E)$. In that case, every TES of the component's behavior is part of a set that is closed under insertion, which means all read events that the robot may do in between two events observe a battery level greater than 0Wh. The behavior property enforces the following safety principle: had there been a violating behavior (i.e., a run where the battery has no energy), then an underlying TES would have observed it, and hence the behavioral property would have been violated.

Another scenario for the two robots is to consider their coordination in order to have them exchange their positions. Let F be initialized to have robot R_1 at position $(0,0)$ and robot R_2 at position $(5,0)$. The property of exchanging position is a liveness

property defined as:

$$\mathrm{P}_{\mathrm{exch}} = \{\sigma \in \mathit{TES}(E) \mid \{(0,0)_1, (5,0)_2\} \subseteq \mathsf{pr}_1(\sigma)(0) \ \mathit{and} \\ \exists i \in \mathbb{N}. \{(5,0)_1, (0,0)_2\} \subseteq \mathsf{pr}_1(\sigma)(i)\}$$

where $(x, y)_i$ is the position of robot i on the field. It is sufficient for a liveness property to be satisfied for the system to be live, i.e., in the case of $\mathrm{P}_{\mathrm{exch}}$ being satisfied, the two robots eventually exchanged position. However, it may be that the two robots exchange their positions before the actual observation happens. In that case, using a similar behavioral property as for safety property will make sure that if there exists a behavior where robots exchange their positions, then such behavior is observed as soon as it happens.

2.2 Division and conformance

Composition is the act of assembling components to form complex systems. This property is particularly desirable if the underlying parts have different type of specifications (e.g., continuous or discrete), but still need to communicate and interact. Our work in Section 2.1 presents a component model that captures both discrete and continuous changes, for which timed-event streams (TESs) are instances of a component behavior. An observation is a set of events with a unique time stamp. A component has an interface that defines which events are observable, and a behavior that denotes all possible sequences of its observations (i.e., a set of TESs). The precise machinery that generates such component is abstracted away. Instead, we present interaction between components as an algebra on components, and we give a wide variety of user defined operations.

Decomposition is dual to composition, as it simplifies a component behavior by removing some of its parts. Decomposition is interesting in two ways: it gives insight on whether a system is composite of a specific component, and it returns a subsystem that, in composition with that component, would give back the initial system. Decomposition is not unique, and may induce a cost or a measure, i.e., a component A may be seen as a product $B \times C$ or $B \times D$ with $C \neq D$. While the qualitative behavior may not change, i.e., the set of sequences of observations stays the same, the substitution of a component with another may somehow *improve* the overall system, e.g., by enhancing its efficiency. For instance, running time is often omitted when specifying systems whose behavior is oblivious to time itself. However, in practice, the time that

a program takes to process its inputs matters. Thus, a component may be substituted with a component exposing the same behavior but running faster. Other criteria such as the size of the implementation, the cost of the production, procurement, maintenance, etc., may be considered in changing one component for another. In this paper, we also consider an orthogonal concern: the cost of *coordination*. Intuitively, the cost of coordination captures the fact that events of two components are tightly related. For example, if two events are related, the occurrence of an event in one component implies the occurrence of some events in another component. While such constraints are declarative in our model, their implementation may be costly. Thus, the relation between observable events of two components may increase the underlying cost to concurrently execute those two components. Finally, having operation to study system decomposition brings alternative perspective on fault detections and diagnosis [50].

Formally, we extend our algebra of components [62] with a new type of operator: a division operation. Division intuitively models decomposition, and acts as an inverse composition operation. Practically, the division of component A by component B returns one component C from all the components D such that $A = B \times D$. Different cost models give rise to different operations of division. We abstractly reason about cost using a partially ordered set of components and show that, for some orders, the set of candidates naturally gives rise to a maximal (minimal) element.

As a running example, we consider a set of robots moving continuously on a shared field. We use the operation of division to specify desirable updates that would prevent robots from interfering with other robots. We also apply division to find simpler components that, if used, would still preserve the entire system behavior. We finally specify the necessary coordination for the robots to self sort on the field.

2.2.1 Divisibility and quotients

Consider two components B and C, and a product $\times$ over components, modelling the interaction constraints on B and C. Then, the composite expression $A = C \times B$ captures, as a component, the concurrent observation of components C and B under the interaction modelled by $\times$. Consider a component D such that $C \times B = D \times B$. If D is different from C, then the equality states that the result of D interacting with B is the same as C interacting with B. Consequently, in this context, component C can be replaced by component D while preserving the global behavior of A.

In general, a component D that can substitute for C is not unique. The set of alternatives for D depends, moreover, on the product $\times$, on the component B,

and on the behavior of A. A 'goodness' measure may induce an order on this set of components, and eventually give rise to a *best* substitution. More generally, the problem is to characterize, given two components A and B and an interaction product $\times$, the set of all C such that $A = C \times B$.

The divisibility of a component A by a component B under product $\times$ captures the possibility to write A as a product of B with another component.

Definition 19 (Right (left) divisibility)**.** *A component A is right (respectively, left) divisible by B under the product $\times$ if there exists a component C such that $B \times C = A$ (respectively, $C \times B = A$).*

A is *divisible* by B under $\times$ when A is both left and right divisible by B under $\times$. Intuitively, the set of witnesses for divisibility, contains all the components that, if taken in a product (under the same interaction signature) with the divisor, yield the dividend. Such witnesses are called *quotients*.

Definition 20 (Right (left) quotients)**.** *The right (respectively, left) quotients of A by B under the product $\times_\Sigma$, written $A/^*_\Sigma B$ (respectively, $A \backslash^*_\Sigma B$), is the set $\{C \mid B \times_\Sigma C = A\}$ (respectively, $\{C \mid C \times_\Sigma B = A\}$).*

If $\times_\Sigma$ is commutative, then $A/^*_\Sigma B = A\backslash^*_\Sigma B$, in which case we write ${}_\Sigma\frac{A}{B}{}^*$. We define left (right) division operators that pick, given a choice function[10], the best element from their respective sets of quotients as their quotients.

Example 25. *Consider a robot that performs 5 moves, and then stops. Each move consumes some energy, and the robot therefore requires sufficient amount of energy to achieve its moves. The product of a robot C with its battery B under the interaction signature Σ is given by the expression $A = C \times_\Sigma B$, where Σ synchronizes a move of robot C with battery B. Note that different batteries behave differently. The set of batteries that would lead to the same behavior is given by the quotients of A by B.*

Definition 21 (Right (left) division)**.** *Let A be divisible by B under $\times_\Sigma$. The right (respectively, left) quotient of A divided by B, under the product $\times_\Sigma$ and the choice function χ over the right (respectively, left) quotients, is the element $\chi(A/^*_\Sigma B)$ (respectively, $\chi(A \backslash^*_\Sigma B)$). We write $A/^\chi_\Sigma B$ (respectively, $A \backslash^\chi_\Sigma B$) to represent the quotient.*

Example 26. *It is usual (e.g., [92]) to consider the greatest common divisor when forming the product of cyber-physical components, so that no observation is missed.*

[10] We assume the axiom of choice [86] and the existence of a function χ that picks an element from a set.

Our operation of division, however, gives an alternative perspective. Let $C(H)$ be a component whose observations have time stamps that are natural multiples of $H \in \mathbb{R}_+$. Then, let $A = C(H_1) \times_\Sigma C(H_2)$. The set of components $\{C(H) \mid A = C(H_1) \times_\Sigma C(H), H \in \mathbb{R}_+\}$ contains all the quotients of A divisible by $C(H_1)$. The selection of the component with the lowest period H would be one choice function for the division of A by $C(H)$ under Σ. When Σ enforces events from $C(H_1)$ and $C(H)$ to occur simultaneously, the product $\times_\Sigma$ projects observation on a common multiple period, and the division returns one component with H as period and whose natural multipliers correspond to the greatest common divisor between integer multiplier in $C(H_1)$ and $C(H_2)$.

If $\times_\Sigma$ is commutative, then $A/^{\chi}_{\Sigma}B = A \setminus^{\chi}_{\Sigma} B$, in which case we denote the division as ${}_\Sigma\frac{A}{B}\chi$.

Example 27 (Lowest element)**.** *One measure to order the set of quotients is to find a component that* is contained *in all the other component behaviors. Indeed, every quotient has the property that, in composition with the divisor, the resulting component equals the dividend. Then, finding a quotient that is contained in all other quotients may be optimal in terms of behavior complexity.*

Let $\mathcal{C}$, the set of right (left) quotients for A divisible by B for product $\times$, is equipped with an ordering such that the lowest element is an element of $\mathcal{C}$, then a function that picks the lowest element can act as a choice function to define the result of the division of A by B. ■

One may consider $\leq$ as a natural ordering on quotients. However, the set of quotients equipped with the containment relation may not have a lowest element. One such example is shown in Table 2.3. Consider A, B, C, and D with $\{0, 1, 2\}$, $\{0, 1\}$, $\{0, 2\}$, and $\{1, 2\}$ as interface, respectively. Using the synchronous composition operation, the TESs τ and η compose with the TES δ to give the TES σ. However, C and D require synchronization on their shared event to compose with B. A smaller component than C and D would be a component F, whose interface is the singleton set containing event 2. However, such component has no shared event with B, and may therefore freely interleave its observations, which does not correspond with observations in A. Thus, F is not an element of the quotients, and C and D have no lower bound in the set of quotients.

We show in the next theorem that a subset of quotients with a shared interface has a lower bound. We discuss how the choice of an interface for a quotient may be guided by some qualitative design choices.

Table 2.3: Counter example for a lowest element in the division of A by B, with C and D two quotients.

	$\sigma : A$	$\tau : B$	$\delta : C$	$\eta : D$
t_1	$\{0, 1, 2\}$	$\{0, 1\}$	$\{0, 2\}$	$\{1, 2\}$
t_2	$\{0, 1, 2\}$	$\{0, 1\}$	$\{0, 2\}$	$\{1, 2\}$
t_3	$\{0, 1, 2\}$	$\{0, 1\}$	$\{0, 2\}$	$\{1, 2\}$
...	...	...	...	

Theorem 2. *Let $\leq$ be the containment relation introduced in Definition 3. Let $\times_\Sigma$ be a commutative, associative, and idempotent product on components, and such that for any two components C and D with the same interface, $C \times_\Sigma D \leq C$. Given A divisible by B under $\times_\Sigma$, any finite subset of quotients sharing the same interface E has a lower bound that is itself a quotient in $A/^*_\Sigma B$.*

Proof. Let $\mathcal{C}(E)$ be a finite subset of the set $\{C \mid C \text{ has interface } E \text{ and } C \in A/^*_\Sigma B\}$. We also write $\times_\Sigma \mathcal{C}(E)$ for the product of all components in $\mathcal{C}(E)$.

For any $C \in \mathcal{C}(E)$, we have

$$\times_\Sigma \mathcal{C}(E) \leq C$$

which makes $\times_\Sigma \mathcal{C}(E)$ a lower bound for $\mathcal{C}(E)$.

Given associativity, commutativity, and idempotency of $\times_\Sigma$, for any $C_1, C_2 \in \mathcal{C}(E)$:

$$\begin{aligned} A &= B \times_\Sigma C_1 \\ A &= B \times_\Sigma C_2 \\ A \times_\Sigma A = A &= (B \times_\Sigma C_1) \times_\Sigma (B \times_\Sigma C_2) \\ A &= B \times_\Sigma (C_1 \times_\Sigma C_2) \end{aligned}$$

which, applied over the set $\mathcal{C}(E)$, gives $A = B \times_\Sigma (\times_\Sigma \mathcal{C}(E))$. Thus, $\times_\Sigma \mathcal{C}(E) \in \mathcal{C}(E)$. [11] □

When conditions of Theorem 2 are satisfied, we write $A/^{\leq, E}_\Sigma B$ for the lower bound of the set of quotients with interface E.

Remark 5. *The operation of division defined by Theorem 2 raises several points for discussion. First, the set of quotients sharing the same interface is structured. Indeed, when the interface is fixed, each finite subset of quotients has a lowest element*

[11] Strictly speaking, closure under finite product does not necessarily imply closure under infinite product. We leave investigating the conditions under which closure under infinite product holds, for future work.

under $\leq$, which makes the definition of a division operator possible. Second, the fact that there is, in general, no minimal element over the set of all quotients reveals the important role that interfaces play in system decomposition. In other words, one may consider another measure to choose *a quotient interface, that is orthogonal to behavior containment (see Section 2.2.4 for a discussion about the cost of coordination).*

We use **1** to denote the component $(\emptyset, TES(\emptyset))$, and **0** to denote the component $(\emptyset, \emptyset)$, that has the empty interface and no behavior.

A component $A = (E_A, L_A)$ is *closed under insertion of silent observations* if, for any $\sigma \in L_A$, and for any silent observation $(\emptyset, t)$ with $t \in \mathbb{R}_+$, and given $i \in \mathbb{N}$ such that $\sigma(i) = (O, t_1)$ and $\sigma(i+1) = (O', t_2)$ with $t_1 < t < t_2$, then there exists $\eta \in L_A$ such that $\sigma(k) = \eta(k)$ for all $k \leq i$, $\sigma(i+1) = (\emptyset, t)$, and $\sigma(k+2) = \eta(k+1)$ for all $k > i$.

In order to reason about components algebraically, we want some properties to hold. For instance, that a component is divisible by itself and the set of quotients contains the unit element.

Lemma 14. *Let A be a component closed under insertion of silent observations, and Σ_{sync} the synchronous interaction signature introduced in Example 4. Then, $\mathbf{1} \in A/^*_{\Sigma_{sync}} A$.*

Proof. For any element $\sigma : A$, and for any $\tau : \mathbf{1}$, we have $(\sigma, \tau) \in R$ and $\sigma[\cup]\tau : A$. Moreover, for any $\sigma : A$, there exists $\tau : \mathbf{1}$ such that $(\sigma, \tau) \in R$ and $\sigma[\cup]\tau = \sigma$. Then, **1** is in the set of quotients of A by A. □ □

Remark 6. *Note that Lemma 14 assumes components to be closed under insertion of silent observations. The reason, as shown in the proof, comes from the product of* **1** *with a component A that may insert silent observations at arbitrary points in time. A consequence of Lemma 14 is the existence of a choice function that can pick, from the set of quotients, the unit component for the division of A by A.*

Example 28. *Let $(R(1,P,T) \bowtie R(2,P,T) \bowtie R(3,P,T)) \times_{\Sigma_{RF}} F_\mu(I)$ be the product of three robot components and a field component with $I = \{1,2,3\}$. Consider the component $P = (E, L)$ with $E = \{read((n,m),i), (n,m)_i \mid n, m \in \mathbb{N}\}$ and $L \subseteq TES(E)$.*

Then, $((R(1,P,T) \bowtie R(2,P,T) \bowtie R(3,P,T)) \times_{\Sigma_{RF}} F_\mu(I))/^{\leq, E'}_{\Sigma_{RF}} P$, with $E' = (E_R(1) \cup E_R(2) \cup E_R(3) \cup E_F(I)) \setminus \{(n,m)_i \mid n, m \in \mathbb{N}\}$, denotes the component that, in composition with P, recovers the initial system. Note that the component resulting from division ranges over the interface E'. As a consequence, all events $(n,m)_i$ have been

hidden in the quotient. Note that the division exists due to the interaction signature Σ_{RF} that imposes simultaneity on occurrence of events read$((n,m),i)$ and $(n,m)_i$. ■

Lemma 15 and Lemma 16 show properties of divisability of components that are similar arithmetic divisability: (1) $n/(n/m)$ is divisible by m, and (2) if n is divisible by m and m is divisible by o then n is divisible by o.

Lemma 15. *Let $\times_\Sigma$ be commutative. Given A divisible by B under Σ and χ a choice function on the set of quotients of A divisible by B, then $B \in A/^*_\Sigma(A/^\chi_\Sigma B)$.*

Proof. If A is divisible by B under Σ and if χ selects one quotient over the set, then $C = A/^\chi_\Sigma B$ is such that $A = B \times_\Sigma C$. By commutativity of $\times_\Sigma$, $A = C \times_\Sigma B$ and $B \in A/^*_\Sigma C$. □

Lemma 16. *Let $\times_\Sigma$ be associative. If A is divisible by B under Σ and B is divisible by C under Σ, then A is divisible by C under Σ.*

Proof. If A is divisible by B under Σ, then there exists D such that $A = B \times_\Sigma D$. If B is divisible by C under Σ, then there exists E such that $B = C \times_\Sigma E$. By substitution, we have $A = (C \times_\Sigma E) \times_\Sigma D$. Using associativity of $\times_\Sigma$, we get $A = C \times_\Sigma (E \times_\Sigma D)$ which proves that A is divisible by C under $\times_\Sigma$. □

2.2.2 Conformance

The criterion for divisibility of A by B, under product $\times$, is the existence of a quotient C such that $B \times C = A$. The equality between $B \times C$ and A makes division a suitable decomposition operator. We can define, a similar operation to describe all components C that *coordinate* B in order for the result to behave in conformance with specification A. In this case, we replace equality with the refinement relation of Definition 2.

Definition 22 (Right (left) conformance)**.** *Component B is right (respectively, left) conformable with component A under $\times$ if there exists a non-empty component C such that $C \times B \sqsubseteq A$ (respectively, $B \times C \sqsubseteq A$).*

Definition 23 (Right (left) conformance coordinators)**.** *The right (respectively, left) conformance coordinators that make B behave in conformance with A under $\times_\Sigma$, denoted as $A \lfloor^*_\Sigma B$ (respectively, $A \rfloor^*_\Sigma B$), is the set $\{C \mid C \times_\Sigma B \sqsubseteq A\}$ (respectively, $\{C \mid B \times_\Sigma C \sqsubseteq A\}$).*

If $\times_\Sigma$ is commutative, then $A \lfloor^*_\Sigma B = A \rfloor^*_\Sigma B$, in which case we write $A \downarrow^*_\Sigma B$. Trivially, every component can be coordinated with the empty coordinator, i.e., the

component $\mathbf{0} = (\emptyset, \emptyset)$. However, the set of coordinators having the same interface is structured and gives ways to define non-trivial coordinators, as in Theorem 3.

Definition 24 (Right (left) coordinator)**.** *Let B be conformable with component A, and let χ be a choice function that selects the best component out of a set of components. The right (respectively, left) coordinator that makes B behave in conformance with A, denoted as $A \lfloor^{\chi}_{\sigma} B$ (respectively, $A \lfloor^{\chi}_{\Sigma} B$), is the component $\chi(A \lfloor^{*}_{\Sigma} B)$ (respectively, $\chi(A \lfloor^{*}_{\Sigma} B)$).*

Example 29 (Greatest element)**.** *One measure to order the set of coordinators is containment. The refinement relation used to define conformance also accepts coordinators that have no behavior at all, and trivially satisfies the behavior inclusion relation. To maximize the observables of the resulting composite behavior set, corresponds to finding the greatest coordinator under the containment relation.*

More generally, if $\mathcal{C}$, the set of right (left) coordinators for B conformable with A under $\times$, is equipped with an ordering such that the greatest element is an element of $\mathcal{C}$, then the function that picks the greatest element can act as a choice function to select the best conformance coordinator of B to behave as A under $\times$. ■

Following the result of Theorem 2, if the interface of the quotient is fixed, then the subset of quotients that have the same interface has a least element with the containment relation introduced in Definition 3. We show in Theorem 3 that a similar result holds for the set of coordinators.

Theorem 3. *Let $\leq$ be the containment relation introduced in Definition 3. Let $\times_{(R,\oplus)}$ be a commutative, associative, idempotent, and monotonic (as in Definition 9) product on components. Given B conformable with A under $\times_{(R,\oplus)}$, any finite subset of coordinators sharing the same interface E has an upper bound that is itself a coordinator in $A \downarrow^{*}_{\Sigma} B$.*

Proof. Let $\mathcal{C}(E)$ be a finite subset of the set $\{C \mid C \text{ has interface } E \text{ and } C \in A \downarrow^{*}_{(R,\oplus)} B\}$. We define the union of two components $A = (E_A, L_A)$ and $B = (E_B, L_B)$, as the component $A \cup B = (E_A \cup E_B, L_A \cup L_B)$. The union of all components in $\mathcal{C}(E)$ is the component $\bigcup \mathcal{C}(E) = (E, \bigcup_{C \in \mathcal{C}(E)} L_C)$ where L_C is the behavior of component C. Moreover, we have that, for any component A, B, C, with B and C sharing the same interface E, $(A \times_{(R,\oplus)} B) \cup (A \times_{(R,\oplus)} C) = A \times_{(R,\oplus)} (B \cup C)$. Indeed let L be the

behavior of $(A \times_{(R,\oplus)} B) \cup (A \times_{(R,\oplus)} C)$ and S be the behavior of $A \times_{(R,\oplus)} (B \cup C)$:

$$\begin{aligned} L &= \{\sigma \oplus \tau \mid \sigma \in L_A, \tau \in L_B, (\sigma,\tau) \in R(E_A,E)\} \cup \\ &\quad \{\sigma \oplus \tau \mid \sigma \in L_A, \tau \in L_C, (\sigma,\tau) \in R(E_A,E)\} \\ &= \{\sigma \oplus \tau \mid \sigma \in L_A, \tau \in L_B \cup L_C, (\sigma,\tau) \in R(E_A,E)\} = S \end{aligned}$$

We show that $\bigcup \mathcal{C}(E)$ is an upper bound for the set of coordinators $\mathcal{C}(E)$. For any $C \in \mathcal{C}(E)$, we have

$$C \sqsubseteq \bigcup \mathcal{C}(E)$$

which implies that $C \leq \bigcup \mathcal{C}(E)$ and makes $\bigcup \mathcal{C}(E)$ an upper bound for $\mathcal{C}(E)$.

Given associativity, commutativity, and idempotency of $\times_{(R,\oplus)}$, for any $C_1, C_2 \in \mathcal{C}(E)$:

$$\begin{aligned} B \times_{(R,\oplus)} C_1 &\sqsubseteq A \\ B \times_{(R,\oplus)} C_2 &\sqsubseteq A \\ (B \times_{(R,\oplus)} C_1) \cup (B \times_{(R,\oplus)} C_2) &\sqsubseteq A \\ B \times_{(R,\oplus)} (C_1 \cup C_2) &\sqsubseteq A \end{aligned}$$

which, applied over the set $\mathcal{C}(E)$, gives $B \times_{(R,\oplus)} (\bigcup \mathcal{C}(E)) \sqsubseteq A$. Thus, $\bigcup \mathcal{C}(E) \in \mathcal{C}(E)$. □ □

Finding a conformance coordinator that makes B behave in conformance with A is looser than finding a quotient for A divisible by B: any quotient of A by B under a product $\times_\Sigma$ is therefore a coordinator that makes B conformable with A. Such quotient-coordinator has the property that it "coordinates" B such that the resulting behavior covers the whole behavior of A.

For some suitable products, Theorem 2 and Theorem 3 state the existence, respectively, of a lowest element in the subsets of quotients and a largest element in the set of coordinators that share the same interface. The synchronous product introduced in Example 4 is one product that satisfies the requirements of each theorem.

2.2.3 Applications of Division

In this section, we consider the robot, field, and protocol components introduced in Examples 19, 20, and 21, together with the synchronous product $\bowtie$ of Example 4 and the product $\times_{\Sigma_{RF}}$ of Example 22. Both products are commutative (Lemma 1 in [62]),

and we therefore omit the right and left qualifiers for division and conformance.

Initial conditions For each robot, we fix the power requirement of a move and the time period T between two observations to be such that a move of a robot during a period T corresponds to a one unit displacement on the field. Then, each move action of a robot changes the location of the robot by a fixed number of units or none if there is an obstacle. We write $R(i)$ for robot $R(i, P, T)$ with such fixed P and T. As an example, the observation $(\{d(i), \mathit{read}(i, (x; y))\}, t)$ followed by the observation $(\{\mathit{read}(i, (x'; y'))\}, t + T)$ gives only few possibilities for $(x'; y')$: either $(x; y) = (x'; y')$, in which case the robot got blocked in the middle of its move, or $(x'; y')$ increases (or decreases) by one unit the x or y coordinates, according to the direction d.

Let the initial state μ of the field be such that $\mu(1) = (3; 0)$, $\mu(2) = (2; 0)$, and $\mu(3) = (1; 0)$, which defines the initial positions of $R(1)$, $R(2)$ and $R(3)$ respectively, and let there be obstacles throughout the field on the 3×2 rectangle from $(0; -1)$ to $(4; 2)$, i.e., for all $(x, y) \in (\llbracket 0; 4 \rrbracket \times \llbracket -1; 2 \rrbracket) \setminus (\llbracket 1; 3 \rrbracket \times \llbracket 0, 1 \rrbracket)$, there exists $i \in I$ such that $\mu(i) = (x, y)_i$. As a result, the moves of each robot are restricted to the inside of the 3×2 rectangle as displayed in Table 2.5.

Approximation of the Field as a Grid

Problem A field component captures in its behavior the continuous responses of a physical field interacting with robots roaming on its surface. The interface of the field contains therefore an event, per object, for each possible position and each possible move. In some cases, however, only a subset of those events are of interest. For instance, we may want to consider only integer position of objects on the grid, and discard intermediate observables. As a result, such component would describe a discrete grid instead of a continuous field, while preserving the internal physics: no two objects are located on the same position. We show how to define the grid as a subcomponent of the field, using the division operator.

Definition of the grid We use division to capture a discrete grid component $G_\mu(I) \leq F_\mu(I)$ contained in the field component $F_\mu(I)$. A grid component has the interface $E_G(I)$, where $E_G(I) \subseteq E_F(I)$ with $(x, y)_i \in E_G(I)$ implies $x, y \in \mathbb{N}$.

We use the component $C = (E_G(I), \mathit{TES}(E_G(I)))$ to denote the free component whose behavior contains all TESs ranging over the interface $E_G(I)$. Then, by application of Theorem 2, we use the least element with respect to $\leq$ of the set of quotients

of $C \times_{\Sigma_{sync}} F_\mu(I)$ divided by $F_\mu(I)$ under Σ_{sync} to define the grid. Thus,

$$G_\mu(I) =_{\Sigma_{sync}} \frac{C \times_{\Sigma_{sync}} F_\mu(I)}{F_\mu(I)}(\leq, E_G(I)) \tag{2.2}$$

which naturally emerges as a subcomponent of the field component $F_\mu(I)$.

Consequences The grid component inherits some physical constraints from the field $F_\mu(I)$, but is strictly contained in the field component. There is a fundamental difference between an approximation of the position as a robot sensor detects, and a restriction of the field to integer positions as in the grid component. In the former, the component reads a value that does not corresponds precisely to its current position, while in the latter, the position read is exact but observable only for integer values.

As a result, the two component expressions $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$ and $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} F_\mu(I)$ restrict each robot behavior in different ways: the grid component allows discrete moves only and the position that a robot reads is the same position as that of an object on the grid, while the field component allows continuous moves but the position that a robot reads is an approximation of the position of the robot on the field. In the sequel, we use the grid component $G_\mu(I)$ instead of the field component.

Updates of components

Problem The interaction signature of a product operator on components restricts which pairs of behaviors are composable. As a consequence, some components may have *more behavior* than necessary, namely the elements that do not occur in any composable pair. An update is an operation that preserves the global behavior of a composite system while changing an operand of a product in the algebraic expression that models the composed system. The goal of such update, for instance, is to remove some behaviors that are not composable or prevent some possible runtime errors. We give an example of such update that replaces a robot component by a new version that removes some of its possibly blocking moves.

Scenario For each robot, we fix its behavior to consist of TESs that alternate between move and reading observations. Moreover, for a robot's period T, and arbitrary $n_i \in \mathbb{N}$, we let $T \times n_i, i \in \mathbb{N}$, represent the timestamp of the i^{th} observation of a TES in its behavior, so long as $n_i < n_{i+1}$. Table 2.4 displays elements of the behavior for each

Table 2.4: Prefixes of four TESs for $R(1)$, $R(2)$, and $R(3)$. For direction d and robot i, we write $d(i)$ instead of $d(i,p)$ since the power p is initially fixed. We omit the set notation as observations are all singletons. We consider $(n_i)_{i\in\mathbb{N}}$ as an increase sequence of natural numbers.

t/T	$\sigma : R(1)$	$\eta : R(1)$	$\tau : R(2)$	$\delta : R(3)$
n_0	$read(1,(3;0))$	$read(1,(3;0))$	$read(2,(2;0))$	$read(3,(1;0))$
n_1	$N(1)$	$W(1)$	$N(2)$	$E(3)$
n_2	$read(1,(3;1))$	$read(1,(2;0))$	$read(2,(2;1))$	$read(3,(2;0))$
n_3	$W(1)$	$W(1)$	$W(2)$	$E(3)$
n_4	$read(1,(2;1))$	$read(1,(1;0))$	$read(2,(1;1))$	$read(3,(3;0))$
n_5	$W(1)$	$\emptyset$	$S(2)$	$\emptyset$
n_6	$read(1,(1;1))$	$\emptyset$	$read(2,(1;0))$	$\emptyset$
n_7	$S(1)$	$\emptyset$	$E(2)$	$\emptyset$
n_8	$read(1,(1;0))$	$\emptyset$	$read(2,(2;0))$	$\emptyset$
n_9	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
...	...	...	...	...

robot. For instance, the TES $\eta : R(1)$ captures the observations resulting from $R(1)$ moving west twice. Note that, in composition with the grid component, the readings may conflict with the actual position of the robot, as some moves may not be allowed due to obstacles on the path.

For instance, given the expression $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$, the TES η is not observable as it is not composable with any of the TESs τ or δ from $R(2)$ and $R(3)$ respectively. We show how to use division to remove all of such behaviors.

Update The replacement for $R(1)$ should preserve the global behavior. We use division to define an update $R'(1)$ of $R(1)$ that removes all elements from its behavior that are not composable with any element from the behavior of $R(2)$ and $R(3)$ under the constraints imposed by the grid.

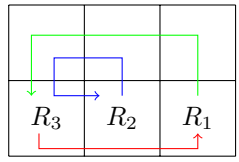
As a result, the component

$$R'(1) = \Sigma_{sync} \frac{(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)}{(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)} (\leq, E_R(1)) \tag{2.3}$$

contains in its behavior all elements ranging over the interface $E_R(1)$ that are composable with elements in the behavior of the dividend component. Note that the set of quotients is filtered on the interface $E_R(1)$, and $R(1)$ trivially qualifies as a quotient. However, $R(1)$ is not minimal as η can be removed from its behavior.

Table 2.5: Prefixes of three TESs for $R(1)$, $R(2)$, and $R(3)$, graphically represented by some trajectories on a grid.

t/T	$\sigma : R(1)$	$\tau : R(2)$	$\delta : R(3)$
n_1	$N(1)$	$N(2)$	$E(3)$
n_2	$W(1)$	$W(2)$	$E(3)$
n_3	$W(1)$	$S(2)$	$read(3,(x_3;y_3))$
n_4	$S(1)$	$E(2)$	$\emptyset$
n_4	$read(1,(x_1;y_1))$	$read(2,(x_2;y_2))$	$\emptyset$
n_5	$\emptyset$	$\emptyset$	$\emptyset$
...	...	...	...



Consequence As a consequence, we defined, using division, an update for component $R(1)$ that removes some elements of its behavior while preserving the global behavior of the composite expression.

Note that the fact that in our example each robot alternates between a move and a read is crucial to remove, by composition, undesired behavior. Indeed, the readings of each robot must synchronize with the location displayed on the grid, and therefore implies that the robot successfully moved. The constraints imposed by the grid coordinate the robot by preventing two robots to share the same location.

Coordination and distribution

Problem Consider the scenario previously described with an additional modification: a robot no longer observes its location after every move, but only at the end of the sequence of moves. The TESs of each robot's behavior are described in Table 2.5, where (x_i, y_i) ranges over possible position readings for robot i. As a result, conflicts between robots may no longer be observable, and the timing of observations may render some incidents of robots blocking each other unobservable. We define a coordinator that makes the system conformant to a global property. As opposed to the division operation, a conformance coordinator may restrict the system behavior to a subset that conforms to a specified property. We consider the following property $P_{sorted}(I)$: "eventually, all the robots get sorted, i.e., every robot $R(i)$ eventually ends on the grid location $(i;0)$.".

Global coordinator We can define, from the sort property, a component as $C_{sorted}(I) = (E_{sorted}(I), L_{sorted}(I))$ whose interface is the union of the interfaces of all robots and the grid, i.e., $E_{sorted}(I) = E_G(I) \cup \bigcup_{i \in I} E_R(i)$, and whose behavior $L_{sorted}(I) \subseteq TES(E_{sorted}(I))$ contains all sequences of moves that make the robots eventually end in their respective sorted grid positions, i.e., $\sigma \in L_{sorted}(I)$ if and only if there exists $t \in \mathbb{R}_+$ such that $(O, t) \in \sigma$ with $(i; 0)_i \in O$ for all $i \in I$. Note that, by construction, the behavior of component $C_{sorted}(I)$ may contain some TESs from the behavior of component $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$, namely ever TESs that satisfies the property.

Consequently, the product of component $C_{sorted}(I)$ with $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$, under the signature Σ_{sync}, defines a component whose behavior contains all elements in the behavior of $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$ that are also in the behavior of $C_{sorted}(I)$. Therefore, if the behavior of component $((R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)) \bowtie C_{sorted}(I)$ is not empty, $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$ is conformant to $C_{sorted}(I)$ and $C_{sorted}(I)$ is a principal coordinator. However, using $C_{sorted}(I)$ as a coordinator requires each component to synchronize, at each step, with every other component. We show how to define a different choice function on the set of coordinators, in order to identify a minimalist form of coordination.

Minimalist coordinator We define a coordinator whose interface is strictly contained in the interface of the global $C_{sorted}(I)$ coordinator. More precisely, we search for a coordinator over the interface of robot $R(1)$ that makes the system $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$ conformant to the property component $C_{sorted}(I)$. First, observe that the set of coordinators

$$(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I) \downarrow^*_{\Sigma_{sync}} C_{sorted}(I)$$

filtered on the interface $E_R(1)$ is empty. Indeed, for any set of timestamp factors n_1, n_2, n_3, and n_4 for the observables of $R(1)$ in Table 2.5, there exists an element from the behavior of $R(2)$ that delays its first action until after n_3, and eventually ends up in a blocking position. As a consequence, there is no coordinator restricted to the events of $R(1)$ that makes $(R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)$ conformant to $C_{sorted}(I)$.

Instead, we consider filtering the set of coordinators with the interface $E_R(1) \cup \{N(2)\}$. In this case, one can find a simple coordinator that makes the set of robots to conform with the sort property. Indeed, every observation $N(1)$ of robot $R(1)$ must occur after an observable $N(2)$ of robot $R(2)$. As a result, such coordinator C_{12}

restricts $R(1)$ to move only after $R(2)$ moves, which results in a composite system conformant to $C_{sorted}(I)$. Thus, the new coordinator C_{12}, in product with $R(1)$, $R(2)$, $R(3)$, and the grid $G_\mu(I)$ satisfies the sort property, i.e., $((R(1) \bowtie R(2) \bowtie R(3)) \times_{\Sigma_{RF}} G_\mu(I)) \bowtie C_{12} \sqsubseteq C_{sorted}(I)$.

Consequence We showed one global coordinator for the set of robots to satisfy the sort property, and one minimalist coordinator over the interfaces of $E_R(1)$ with an event from $E_R(2)$. The minimalist coordinator has an interface strictly included in the global coordinator, which therefore minimizes the amount of interaction among components. In the next section, we discuss the cost of coordination as a possible measure to order a set of conformance coordinators.

2.2.4 Discussion

The operations of division and conformance defined earlier characterize all possible updates and coordinators for a composite system. In general, the set of quotients or coordinators is not a singleton, which then necessitates a choice function to *pick* a component that suits the needs. We saw in Theorem 2 how such a choice function can be defined using an ordering on components, and choosing the least of such component. Intuitively, such choice function prefers a quotient with the least amount of observations. However, Theorem 2 assumes a fixed interface, and does not discuss how to rank components according to their interface. We discuss alternative rankings hereafter in the case of a synchronous product $\bowtie$ of Example 4.

Cost of coordination

Let $A = (E_A, L_A)$ and $B = (E_B, L_B)$ be two components such that the set of quotients of A divisible by B under $\bowtie$, the synchronous product, is non empty. Consider, as well, a component $C = (E_C, L_C)$ in the set of quotients. We discuss alternative scenarios based on the interface of component C.

Consider the case where $E_C \cap E_B \neq \emptyset$. Then, events in the intersection $E_C \cap E_B$ are events for which C and B must perform a simultaneous observation, i.e., with equal time stamp. The size of the intersection $E_C \cap E_B$ therefore characterizes how much coordination should take place among two implementations of components B and C to successfully achieve the synchronous observations. Alternatively, if a component $D = (E_D, L_D)$ is in the set of quotients such that $|E_D \cap E_B| < |E_C \cap E_B|$, the number

of shared events is smaller, which hints at a smaller amount of coordination among components D and B.

In the case that $E_C \cap E_B = \emptyset$ and $E_C \neq \emptyset$, the family of quotients with interface E_C are particularly useful. The fact that the two interfaces are disjoint tells us that A can be decomposed in two components that do not need any coordination. Indeed, as $\bowtie$ only constraints occurrences of shared events, if B and C share no events then they can be run completely independently.

The two cases highlighted above give us some insight on how coordination can be used as a measure to rank components. Note that such ranking is contextual to the dividend and the divisor. Even though C may require more coordination than D to synchronize with B in order to form component A, in the context of A divisible by another component F, the component C may become preferable to D.

The cost of coordination discussed here is orthogonal to efficiency measures discussed in 2. For that reason, the two measures can be combined to first rank components in terms of their interface to minimize the amount of coordination required, and then rank components sharing the same interface in terms of the size of their behavior. More thorough analysis could be done as to compare, from the behavior of each component, how often coordinated event effectively occurs.

Series of division

We defined the operator of division on components. We saw that, under some criteria, division may return a 'better' description of a composite system, i.e., for $A = B \times C$, the division of A by B may return a D better than C while preserving the behavior of A.

Then, one question that follows is that of convergence. Consider the expression $A = B \times C$, and the division of A by B returning a component $C' \neq C$. Symmetrically, the division of A by the new component C' may return a component $B' \neq B$. Repeating the same process, dividing A by B' and so on, gives a sequence of components $C^{(n)}$ and $B^{(n)}$ for the respective n^{th} division. Does the sequence eventually converge to a fixed pair of components?

2.3 Linearization

The algebra that we introduced in Section 2.1 is expressive enough to model a large range of concurrent systems. While the focus of this thesis is on components with

cyber-physical aspects, we can use the algebraic setting to prove useful properties of components with cyber-cyber interactions.

In this section, we consider *order sensitive components*, which are components for which the order of observations matters but not the exact time labels of observations. For that reason, the behavior of such a component is closed under non-uniform stretching time, as long as the order of observations remains the same. We relax the notation throughout this section, and use a sequence $\sigma \in \mathcal{P}_f(\Sigma)^\omega$ to effectively reason about all the TESs $\tau \in TES(\Sigma)$ that have the same order of observations, i.e., $\sigma = \mathsf{pr}_1(\tau)$.

Programming languages like Reo [47] and LUSTRE [24] describe concurrent systems in terms of sequences of *transactions* (or interactions). A transaction is a finite set of actions that occur atomically, which corresponds to a timeless observation, as introduced in Section 2.1. That is, a transactions *occurs* if and only if all the actions in the set occur, without the interference of any other action in between. Atomicity captures the notion of the *all-or-nothing* behavior and the possibility of interleaving of independent events only (see Section 2.1). A prime example of a transaction in concurrent systems is a barrier synchronization, which allows a group of processes to proceed only when all of them reach a particular local state. We refer to an infinite sequence of transactions as a transactional trace, and refer to a set of transactional traces as a transactional behavior. Note that a transactional behavior is independent of the time at which the transaction occurs, and records the order of occuring transactions only. We use $\mathbb{T}$ to denote the set of transactional behaviors. Transactional behaviors are particularly suitable to describe *what* behavior is acceptable in a concurrent system, by declaring all sequences of transactions that are allowed. A transactional component denotes a transactional behavior restricted to a set of actions that we refer to as its interface. Transactional components are therefore a subclass of components as introduced in Section 2.1.

A transactional component is *linear* if and only if every transaction in this component is a singleton set or the empty set. Programming languages without syntax for such transactions, like [23, 80, 11], describe concurrent systems as linear (transactional) components. Linear components describe sequential behaviors and are particularly suitable to describe *how* the behavior of a component is generated, since at most one action happens at any given time. We use $\mathbb{L}$ to denote the set of transactional behaviors.

When two processes that have shared actions in their interfaces are composed, the occurrence of a shared action in their behaviors must coincide. This coincidence is understood as synchronous communication between the two processes. The concept

of (a)synchronous communication is orthogonal to transactional or linear behaviors. In this section, we deal exclusively with synchronous communication. To construct complex concurrent systems out of simpler components, we equip components with a composition operator that captures synchronous communication.

The design of a specification of a concurrent system is easier using transactional components [4], while its execution is better captured by linear components. However, in practice, a sequential processor can do at most one action at a time, and the number of actions that can happen at the same time is bounded by the number of parallel processors. Consequently, an implementation of a synchronous component on a sequential machine necessitates to linearize transactions to sequences of actions. A linearization of a synchronous component is valid if it preserves the *all-or-nothing* semantics of its transactions, and linearization of the product is the product of linearizations. The problem is therefore to characterize what linearization from transactional components in $\mathbb{T}$ to linear components in $\mathbb{L}$ are valid. Intuitively, the linearization of a transactional component is valid if the behavior of the resulting components preserves the intended semantics of the behavior of the initial transactional component.

Moreover, in the case of concurrent and distributed systems, compositionality is an important feature that allows one to run software in parts, and assemble the compiled parts at runtime. Besides static composition, dynamic composition is required for, e.g., dynamic updates, modular compilation, or reconfiguration. Semantically, runtime composition is captured by the composition operation $\otimes$ in $\mathbb{L}$. After defining transactional components in Section 2.3.2, we characterize valid linearizations in Section 2.3.4 and give two practical instances, one that runs every component in lockstep, and one that allows for interleaving of independent transactions. More particularly, we require that a sequence after linearization contains all events of a transactional behavior and preserve some ordering among dependent events.

2.3.1 Dependency and concurrency

We fix a (possibly infinite) set Σ of *primitive actions*, and write $\tau \notin \Sigma$ to denote an *internal action*. We write $\Sigma_\tau = \Sigma \cup \{\tau\}$ to denote the set of actions. We do not make any assumptions on the structure of primitive actions. For instance, Σ may consist of actions a_c of the meaning "the traffic light indicates color c", or Σ may consist of actions $a_{p,d}$ of the meaning "port p fires with data d". We use regular expressions to denote a set of sequences of actions. Given a set X, with $a, b \in X$, the expressions $a + b$, ab, a^* respectively denote the sets of sequences $\{a, b\}$, $\{ab\}$, and $\{a, aa, aaa, ...\}$.

Later on, we use regular expressions over a carrier that is the finite power set of X: we should not confuse the set notation from the terms of the carrier and the set notation from the set of sequences that the expression denotes.

Some actions are dependent. For instance, the statement "port p fires with data d always happens before port p fires with data e" induces a dependency relation between $a_{p,d}$ and $a_{p,e}$. We define dependency as follows.

Definition 25. *Dependency is a partial order $\leq$ on Σ_τ, such that τ is not related to any primitive action in Σ.*

In the sequel, we consider a fixed but arbitrary dependency $\leq$ and say that a *happens before* b if $a \leq b$. Dependency induces a symmetric, reflexive dependency relation

$$D_\leq = \{(a, b) \in \Sigma^2 \mid a \leq b \text{ or } b \leq a\},$$

which brings us to the realm of Mazurkiewicz traces [67]. Actions a and b are *dependent* if $(a, b) \in D_\leq$ and are *independent* if $(a, b) \notin D_\leq$. Note that, without loss of generality, we can make the partial order strict and remove the elements (a, a) from the dependency relation.

Remark 7. *Consider a client that can send requests a and b. Suppose that $(a, b) \notin D_\leq$. Then, a and b are independent (e.g., one request does not require the other request) and the client may perform the requests in parallel.* ◊

The dependency relation allows us to permute independent actions. In the case of finite sequences, it suffices to define a trace equivalence as the smallest equivalence relation that allows commutation of consecutive independent actions. However, such a definition would allow only finitely many commutations, which means that $(ab)^\omega$ and $(ba)^\omega$ are not trace equivalent, whenever a and b are independent. Following Gastin [37], we define trace equivalence by means of (infinite) dependency graphs.

Definition 26. *The dependency graph $\Gamma_\leq(s)$ of a sequence $s \in \Sigma_\tau^\omega$ of actions is a graph (V, E) with vertices V and edges E defined as*

$$V = \{(a, i) \mid a \in \Sigma_\tau, 0 \leq i < |s|_a\}$$
$$E = \{(a, i) \to (b, j) \mid (a, b) \in D_\leq \text{ and the } i\text{th } a \text{ occurs before the } j\text{th } b \text{ in } s\}$$

where $|s|_a \in \mathbb{N} \cup \{\infty\}$ is the number of occurrences of action $a \in \Sigma_\tau$ in s.

We define trace equivalence (modulo dependency $\leq$) as the kernel of $\Gamma_{\leq}$:

$$\equiv_{\leq} \;=\; \ker(\Gamma_{\leq}) \;=\; \{(u,v) \mid \Gamma_{\leq}(u) = \Gamma_{\leq}(v)\}.$$

In the sequel, we drop the subscript $\leq$, if there is no danger of confusion. For a sequence $s \in \Sigma_\tau^\omega$, we write $[s] = \{r \in \Sigma_\tau^\omega \mid r \equiv s\}$ for the equivalence class of s. For a set $L \subseteq \Sigma_\tau^\omega$, we write $[L] = \{r \in \Sigma_\tau^\omega \mid s \in L, r \equiv s\}$ for the closure of L under trace equivalence.

Remark 8. *The most primitive verification tool of a software engineer is the print statement. Let Σ denote the set of all possible results of every print statement occurring in a program. If the program is sequential, its observable behavior can be accurately expressed as a set $L \subseteq \Sigma^\infty$ of possibly infinite sequences of observations. The set L contains all relevant observable information of a sequential system.*

If the software is concurrent, distinct statements may print simultaneously. To prevent unreadable output, the print statements acquire exclusive access to the log file or console before printing. While such linearization of observations restores the readability of the output, it hides the fact that the program is concurrent. Hence, the set of sequences of observations does not *contain all relevant information of a concurrent system.* $\Diamond$

2.3.2 Transactional and linear components

We fix a collection $T \subseteq \mathcal{P}_f(\Sigma)$ of finite sets of primitive actions, called *transactions*. A transaction $A \in T$ is atomic, as defined in Section 2.1, such that all primitive actions in A occur simultaneously, and no other dependent action can interleave. The empty transaction $\emptyset$ is related to the internal τ step as we will see later. Note that some actions in Σ are conflicting, such as $a_{p,d}$ and $a_{p,e}$ from Section 2.3.1, for distinct data $d \neq e$. Since conflicting actions never occur in the same transaction, we generally have $T \neq \mathcal{P}_f(\Sigma)$ and $\emptyset \in T$.

A *transactional trace* is an infinite sequence $A_0 A_1 A_2 \cdots \in T^\omega$ of transactions. A *transactional behavior* is a set $K \subseteq T^\omega$ of transactional traces. The composition operator on transactional behaviors $K \in \mathcal{P}(T^\omega)$ is intersection, $\cap$, of sets. Let $I \subseteq \Sigma$ be a set of primitive actions. Consider the largest[12] symmetric relation $\sim_I$ on T^ω that

[12]This largest relation exists, because the union of all $\sim_I$ that satisfy Equation 2.4 satisfies Equation 2.4.

satisfies

$$\forall r, s \in T^\omega \qquad r \sim_I s \quad \Rightarrow \quad r(0) \cap I = s(0) \cap I \text{ and } r' \sim_I s', \tag{2.4}$$

where $r'(i) = r(i+1)$ and $s'(i) = s(i+1)$, for all $i \geq 0$. A transactional behavior $K \subseteq T^\omega$ is $\sim_I$*-closed*, whenever $r \sim_I s \in K$ implies $r \in K$, for all $r, s \in T^\omega$.

Given a transactional behavior K, a *transactional component* is a pair (I, K) with $I \subseteq \Sigma$ and such that K is $\sim_I$-closed. We call I its *interface*.

Example 30 (Sync channel)**.** *A Sync channel is a transactional component with* $I = \{a, b\}$ *as interface containing an input port* a *and an output port* b*. Whenever the Sync fires its input port, it simultaneously fires its output port.*

Formally, let $\Sigma = \{a, b\}$ *consist of two actions* a *and* b *that correspond with firing of the input and output ports of the Sync channel, respectively. The behavior of the Sync is defined as the set* $(\{a, b\} + \emptyset)^\omega$*.*

In a larger context, such as $\Sigma = \{a, b, c\}$*, for some other port* c*, behavior for the Sync component extends to* $(\{a, b, c\} + \{a, b\} + \{c\} + \emptyset)^\omega$*, because we assume that components are closed under addition/removal of actions outside of their interface. That is, the set* $(\{a, b, c\} + \{a, b\} + \{c\} + \emptyset)^\omega$ *is* $\sim_I$*-closed.* ◊

Remark 9. *There is a subtle difference between concurrent actions and transaction. Concurrency is about independence of actions such that they* may *happen simultaneously. A transaction is about timing of actions such that they* will *happen atomically, which may also be simultaneous. Since the two terms are both referring to actions that may happen at the same instance, it is easy to confuse the terms. A transaction makes sense only in a concurrent setting: actions are atomic and are therefore concurrent. However, concurrent actions are not necessarily atomic.* ◊

Example 31. *Let* C_1 *and* C_2 *be two Sync components, with interface* $\{a, b\}$ *and* $\{b, c\}$*, respectively. The behavior of the composition of* C_1 *and* C_2 *consists of all behaviors in* C_1 *that are also behaviors in* C_2 *(composition is intersection). By modelling atomicity of actions with transactions one gets transitivity of atomicity for free. Indeed, if* a *and* b *are atomic and* b *and* c *are atomic, then* a *and* b *occur within the same transaction and* b *and* c *occur within the same transaction. Hence,* a *and* c *eventually occur within the same transaction in the composition of* C_1 *and* C_2*, which means that they are atomic.* ◊

A *linear trace* is a special kind of transactional trace, where every element is either a singleton action, or the empty set. In order to simplify notation, we identify a

sequence of actions $a_0a_1a_2\cdots \in \Sigma_\tau^\omega$ with the linear trace $A_1A_2A_3\cdots \in \mathcal{P}(\Sigma)^\omega$, where each action a_i corresponds to the singleton transaction $\{a_i\}$ if $a_i \neq \tau$, and to the transaction $\emptyset$ otherwise. A *linear behavior* is a set $L \subseteq \Sigma_\tau^\omega$ of linear traces, and a *linear component* is a pair (I, K) of an interface I and a linear behavior K that is $\sim_I$ closed.

We consider an arbitrary associative, commutative, idempotent composition operator $\otimes$ on linear components. Typically, intersection of sets, $\cap$, is used as composition operator but see Section 2.3.4 for an example of a different composition operator.

Linearization Every transactional behavior has one or more equivalent linear behaviors that are related to it via linearization. The linearization is defined hierarchically, from transactions to traces and behaviors. Each level "lifts" the definition of linearization to sequences of transactions, and to sets of sequences of transactions.

The linearization of a transaction results in a set of sequences of actions. Our fixed dependency of actions, $\leq$, restricts the order in which the actions of a transaction can linearize. Let $A \subseteq \Sigma$ be a finite set of primitive actions. An element $a \in A$ is *minimal* in A iff $x \leq a$ implies $x = a$, for all $x \in A$. We write $\min(A)$ for the set of all minimal elements of A.

Let λ denote the empty sequence. The set of linearizations $\ell(A) \subseteq \Sigma_\tau^*$ of a transaction $A \in T$ with respect to dependency is defined inductively on the size of A as $\lambda \in \ell(\emptyset)$, and if $u \in \ell(A)$ and $a \in \min(A \cup \{a\}) \setminus A$, then $\tau u \in \ell(A)$ and $au \in \ell(A \cup \{a\})$. One reason to allow arbitrary interleaving of τ-actions is that it allows us to encode transactions with an explicit terminating τ-step.

Example 32. *We have* $\ell(\emptyset) = \tau^*$, $\ell(\{a\}) = \tau^* a \tau^*$, *and*

$$\ell(\{a,b\}) = \begin{cases} \tau^* a \tau^* b \tau^* & \text{if } a < b, \\ \tau^* b \tau^* a \tau^* & \text{if } b < a, \\ \tau^*(a\tau^* b + b\tau^* a)\tau^* & \text{otherwise,} \end{cases}$$

where $a \neq b$ *and* τ^* *is the Kleene star operator applied on* τ, *and* $+$ *is union.* $\Diamond$

Definition 27 (Linearization)**.** *The linearization relation is coinductively defined as the largest*[13] *relation* $\rightsquigarrow \subseteq T^\omega \times \Sigma_\tau^\omega$ *that satisfies one of the following conditions. If* $A_0A_1A_2\cdots \rightsquigarrow a_0a_1a_2\cdots$, *then either*

[13]The largest relation exists, because the union of all $\rightsquigarrow$ that satisfy Definition 27 satisfies Definition 27

1. *for all $i \geq 0$, we have $A_i = \emptyset$ and $a_i = \tau$; or*

2. *for some $i, j \geq 0$, $A_i \neq \emptyset$, $a_0 \cdots a_j \in \ell(A_0)$, and $A_1 A_2 \cdots \rightsquigarrow a_{j+1} a_{j+2} \cdots$.*

The first item of Definition 27 considers the case of a sequence with the empty transaction only. In this case, the resulting linear behavior is the singleton set with τ actions only.

We explain informally the second item in Definition 27. The index i and the condition $A_i \neq \emptyset$ is to exclude item (1). The index j and the sequence $a(0)...a(j)$ is a linearization of A_0, which is the head of the sequence of transactions. The last part (i.e., $A_1 A_2 ... \rightsquigarrow a(j+1)a(j+2)...$) is the coinductive definition.

An empty transaction $A_i = \emptyset$, for $i \geq 0$ of a sequence $A_0 A_1 A_2 \cdots \in T^\omega$ is trailing iff $A_j = \emptyset$, for all $j \geq i$. Linearization ignores non-trailing empty transactions, as they can match with the empty sequence of actions. However, linearization recognizes trailing empty transactions and relates them to τ^ω.

Example 33. *We have $\emptyset^\omega \rightsquigarrow \tau^\omega$. Suppose that $a < b$. Then, $\{a, b\}^\omega \rightsquigarrow s$ if and only if $s \in (\tau^* a \tau^* b)^\omega$. Also, $(\{a, b\}\emptyset)^\omega \rightsquigarrow s$ if and only if $s \in (\tau^* a \tau^* b)^\omega$. Thus, linearization ignores non-trailing empty transactions. We also have $(\{a\}\{b\})^\omega \rightsquigarrow s$ if and only if $s \in (\tau^* a \tau^* b)^\omega$. Hence, linearization also confuses the transaction $\{a, b\}$ with the sequence of transactions $\{a\}\{b\}$.* ◊

For a transactional behavior $K \subseteq T^\omega$, we define $K{\rightsquigarrow} = \{s \in \Sigma_\tau^\omega \mid \exists \alpha \in K, \alpha \rightsquigarrow s\}$ to be the linear component that contains all linearizations of sequences from K. The set of linearizations $K{\rightsquigarrow}$ of K is generally not closed under trace equivalence.

Example 34. *Consider the actions $\Sigma = \{a, b, c, d\}$, with dependency $a < b$ and $c < d$. Let $\alpha = \{a, b\}\{c, d\}\emptyset^\omega$, $s = abcd\tau^\omega$, and $s' = ac\tau bd\tau^\omega$. Since τ is not related to b or d by Definition 25, we have $s \equiv s'$ (see Definition 26). Furthermore, we have $\alpha \rightsquigarrow s$, while $\alpha \not\rightsquigarrow s'$. Therefore, $\{\alpha\}{\rightsquigarrow}$ is not closed under trace equivalence.* ◊

Definition 28 (Weak linearization)**.** *The weak linearization relation is the composition $\rightsquigarrow\equiv$ of linearization and trace equivalence.*

That is, $\alpha \rightsquigarrow\equiv s$ if and only if $\alpha \rightsquigarrow s' \equiv s$, for some $s' \in \Sigma_\tau^\omega$. By construction, $K{\rightsquigarrow}\equiv = \{s \in \Sigma_\tau^\omega \mid \exists \alpha \in K, \alpha \rightsquigarrow\equiv s\} = [K{\rightsquigarrow}]$ is closed under trace equivalence.

2.3.3 Problem statement: compositional linearization

Let $T \subseteq \mathcal{P}_f(\Sigma)$ be a set of transactions over the set of actions Σ. Let $\mathbb{T} = \mathcal{P}(T^\omega)$ be the space of transactional behaviors with $\cap$ as composition operator. Let $\mathbb{L} \subseteq \mathbb{T}$

be the space of linear behaviors with an arbitrary commutative, associative, idempotent product $\otimes$ as composition operator. As introduced in previous subsection, the linearization relates transactional behaviors to linear behaviors.

Let $\varphi : \mathbb{T} \to \mathbb{L}$ be a linearization function. Then, φ is a compositional linearization if and only if φ is a homomorphism, i.e., for all behaviors $K_1, K_2 \in \mathbb{T}$, $\varphi(K_1 \cap K_2) = \varphi(K_1) \otimes \phi(K_2)$. Not all compositional linearization give rise to a *useful* linearization. As an example, take the trivial linearization that maps every transactional behaviors to the singleton set containing the sequence of empty transaction. While being compositional, such linearization does not preserve the intended semantics. Instead, we consider *valid* linearizations.

Definition 29 (Valid linearization)**.** *Let $\varphi : \mathbb{T} \to \mathbb{L}$ be a linearization. Then, φ is* valid *if:*

1. *φ is compositional, i.e., for all $K, K' \in \mathbb{T}$, $\varphi(K \cap K') = \varphi(K) \otimes \varphi(K')$; and*
2. *$\rightsquigarrow\!\equiv$ is total and surjective on $K \times \varphi(K)$ for all $K \in \mathbb{T}$.*

The first item in Definition 29 implies that φ preserves composition, which allows us to linearize in parts and assemble later (possibly at run time). This is particularly useful for the inclusion of closed-source third-party software, which is precompiled by its vendor. Furthermore, first item in Definition 29 can be used to speed up application of updates, since parts that are not changed do not need to be linearized again.

The second item in Definition 29 means that, for every behavior $K \in \mathbb{T}$, every trace in $\varphi(K)$ is a linearization of some trace in K, and every trace in K has some linearization in $\varphi(K)$. In other words, the second item in Definition 29 asserts that a valid linearization does actually 'linearize'. Moreover, the equivalence relation $\equiv$ enables commutation of events that are independent, which corresponds to the second clause of the definition of atomicity of a transaction, as defined in Section 2.1.

Remark 10. *Delinearization is the translation of a linear behavior to a transactional behavior. If a linearization φ is injective, it naturally comes with a delinearization φ^{-1} that, for those linear behaviors in the image of φ, gives back a transactional behavior.* $\Diamond$

Note that $\rightsquigarrow$ as defined in Section 2.3.2 is not compositional for $\otimes = \cap$, and therefore not valid. As an example, let $K_1 = (\emptyset\{a\})^\omega$ and $K_2 = (\{a\}\emptyset)^\omega$. Then, $K_1 \cap K_2 = \emptyset$ while $K_1{\rightsquigarrow} \cap K_2{\rightsquigarrow} = (\tau^* a \tau^\omega) \cap (\tau^* a \tau^\omega) = \tau^* a \tau^\omega$. In the next section, we characterize all valid linearizations, and we give two practical instances.

2.3.4 Valid linearizations: lock step and interleaving

As defined in Definition 29, a *valid linearization* is a function that maps a transactional behavior to a linear behavior while preserving the compositional structure. We seek a morphism whose co-domain is a linear behavior from the set of transactions of its domain.

The lock-step linearization is a valid linearization which we present first. Each linearization is delineated with a special τ symbol. The resulting linear behavior has, for each trace, a τ symbols that delineates every transaction, and is therefore homomorphic with set intersection as the composition operation. While intuitive, such linearization is not efficient and requires every component in the intersection to run in lock-step.

We give another instance of a valid linearization, where we relax the lock-step behavior to tolerate some interleaving of independent actions.

Lock-step linearization A straightforward example of a valid linearization is based on synchronous rounds. Here, we use the τ action (also the empty transaction) to indicate the end of a round. This allows us to reconstruct the transactions from a sequence of actions. To be precise, we define *grouping* coinductively as follows.

Definition 30 (Grouping)**.** *The grouping relation is the largest*[14] *subset* $G \subseteq \Sigma_\tau^\omega \times T^\omega$*, such that* $(a_0 a_1 \cdots, A_0 A_1 \cdots) \in G$ *implies*

1. $a_0 = \tau$*,* $A_0 = \emptyset$*, and* $(a_1 a_2 \cdots, A_1 A_2 \cdots) \in G$*; or*
2. $a_0 \in \min(A_0)$ *and* $(a_1 a_2 \cdots, (A_0 \setminus \{a_0\}) A_1 \cdots) \in G$

Example 35. *If* a *and* b *are independent, then* $(ab\tau\tau ba\tau^\omega, \{a,b\}\emptyset\{a,b\}\emptyset^\omega) \in G$*. If* a *and* b *are such that* $a < b$*, then* $(ab\tau^\omega, \{a,b\}\emptyset^\omega) \in G$ *but* $(ba\tau^\omega, \{a,b\}\emptyset^\omega) \notin G$*.* ◊

Lemma 17. *G is a functional relation.*

Hence, we find a partial function $g : \Sigma_\tau^\omega \rightharpoonup T^\omega$ with $g(a_0 a_1 \cdots) = A_0 A_1 \cdots$ if and only if $(a_0 a_1 \cdots, A_0 A_1 \cdots) \in G$. The domain of g consists of all sequences $s \in \Sigma_\tau^\omega$, such that τ occurs infinitely often in s, and every primitive action occurs at most once between consecutive τ actions. We define the linearization based on synchronous rounds as the preimage of grouping g.

[14] Again, the largest relation exist.

Definition 31. *For every transactional behavior $K \subseteq T^\omega$, we define*

$$g^{-1}(K) = \{s \in dom(g) \mid g(s) \in K\}$$

It is straightforward to check that g^{-1} is a valid linearization, if we take intersection $\cap$ as the composition operator $\otimes$ on $\mathcal{P}(\Sigma_\tau^\omega)$.

Theorem 4. *g^{-1} is a valid linearization, for $\otimes = \cap$.*

Although Theorem 4 shows that the linearization based on synchronous rounds is valid, the resulting linear behaviors are locking the execution into strict rounds. As a result, independent actions can only swap within a transaction but not over sequences of transactions.

Example 36 (Parallel Syncs)**.** *Consider a Sync channel K_1 from a to b and a Sync channel K_2 from c to d. Set $\Sigma = \{a, b, c, d\}$ with $a < b$ and $c < d$. The composition of K_1 and K_2 is the behavior*

$$K_1 \cap K_2 = (\{a, b, c, d\} + \{a, b\} + \{c, d\} + \emptyset)^\omega.$$

Then, $g^{-1}(K_1 \cap K_2)$ equals

$$(a(bcd + c(bd + db))\tau + c(dab + a(bd + db))\tau + ab\tau + cd\tau + \tau)^\omega$$

Now consider the prefix acb of a behavior in $g^{-1}(K_1 \cap K_2)$. Ignoring the c from the second Sync K_2, we see that acb completes an $\{a, b\}$ transaction of the first Sync K_1. One would expect that the transactions $\{a, b\}$ and $\{c, d\}$ are independent, because they are disjoint and their respective actions are not related by the partial order. However, $g^{-1}(K_1 \cap K_2)$ dictates that the second Sync K_2 must complete its $\{c, d\}$ transaction before anything else can happen. Hence, first Sync K_1 can accept new input only after $acbd\tau$. Consequently, the concurrency of the transaction $\{a, b\}$ and $\{c, d\}$ is weaker than expected: while actions a and b are independent with actions c and d, the two transaction $\{a, b\}$ and $\{c, d\}$ cannot interleave arbitrarily.

We want to design a valid linearization φ, such that, for instance, the sequence $(acdcdb\tau)^\omega$ is part of $\varphi(K_1 \cap K_2)$. ◊

Interleaving tolerant linearization To avoid the oversynchronization in Theorem 36, we must allow traces that minimize the explicit τ action, i.e., the τ inserted after every round in the lock-step linearization. Observe that $g^{-1}(K) \subseteq dom(g)$, for

every behavior $K \subseteq T^\omega$. The set $dom(g)$ consists of sequences over Σ_τ that contain infinitely many τ actions and between any two τ actions, every primitive action happens at most once.

Recall the linearization relation $\rightsquigarrow$ from Section 2.3.2 and from Section 2.3.1 that $[L]$ is the closure of $L \subseteq \Sigma_\tau^\omega$ with respect to trace equivalence $\equiv$. Consider the map φ defined, for all behavior $K \subseteq T^\omega$, as

$$\varphi(K) = g^{-1}(K) \cup ([K\rightsquigarrow] \setminus dom(g)),$$

where $[K\rightsquigarrow]$ is the closure modulo trace equivalence of the set of all linearizations of behaviors from K. Intuitively, the linearization φ adds to the image of g^{-1} some linear behaviors that are not in the domain of the grouping. Observe that φ is injective, since

$$g(\varphi(K) \cap dom(g)) = g(g^{-1}(K)) = K.$$

Consider the composition operator $\otimes$ defined, for behaviors $L_1, L_2 \subseteq \Sigma_\tau^\omega$, as

$$L_1 \otimes L_2 = L_1 \cap L_2 \cap [L_1 \cap L_2 \cap dom(g)]. \tag{2.5}$$

Lemma 18. *For $L \subseteq dom(g)$, we have $[L] = [g(L)\rightsquigarrow]$.*

Theorem 5. *φ is valid, with $\otimes$ defined in Equation 2.5.*

Example 37 (Parallel Syncs with concurrency)**.** *Consider the Sync channels as in Example 36. Then, the linear behavior $\varphi(K_1 \cap K_2)$ contains runs like $acbababd\tau^\omega$. Although actions a and b are independent to actions c and d, the linearization g^{-1} forbid arbitrary interleaving of transactions $\{a, b\}$ and $\{c, d\}$. As a result, the linearization φ allows for such arbitrary interleaving.* ◊

2.4 Related work and future work

(De)composition. In [27] , the authors present a declarative and an operational theory of components, for which they define a refinement relation and compositionality results for some composition operators. Our work is related as it aims for similar results, but for the case of Cyber-Physical systems. Thus, instead of having input and output actions, components have timed observations, and composability relations. We present, as well, quotient operation on components, and show how it can be used to synthesize coordinating CPSs.

In [76], the authors consider the problem of decomposition of constraint automata. This work provides a semantic foundation to prove that the construction in [76] is a valid division.

In [71], the authors consider the problem of decomposition in process algebra for the parallel operator. The notion of prime process is introduced, and the unicity of decomposition of a process as a parallel composition of primes is posed. The problem is answered for different types of congruences on processes. The work has been extended for decomposition of processes in the π-calculus in [33].

Algebra, co-algebra The algebra of components described in this paper is an extension of [62]. Algebra of communicating processes [36] (ACP) achieves similar objectives as decoupling processes from their interaction. For instance, the encapsulation operator in process algebra is a unary operator that restricts which action to occur, i.e., $\delta_H(t \parallel s)$ prevent t and s to perform actions in H. Moreover, composition of actions is expressed using communication functions, i.e., $\gamma(a, b) = c$ means that actions a and b, if performed together, form the new action c. Different types of coordination over communicating processes are studied in [20]. In [12], the authors present an extension of ACP to include time sensitive processes.

The modeling of component's interaction using co-algebraic primitives is at the foundation of the Reo language [8]. In [18], the question of separation of components into two sub-components is addressed from a co-algebraic perspective.

The interaction signature that parametrizes algebraic operator is related to the synchronization algebra in [90]. A synchronization algebra relates events labeling edges of a synchronization tree. The product of two synchronization trees is therefore parametrized by the underlying synchronization algebra on its events. Our interaction signature generalizes that of synchronization algebra in the sense that composability relations are defined at three levels: on TESs, on observations, or on events. We study mechanisms to lift composability relations from events to observations and to TESs, therefore proving the algebraic properties of the parametrized product given algebraic properties of the underlying composability relation.

In [38], the authors consider a monadic semantics for hybrid programs. The hybrid monad captures the continuous behavior of a hybrid program and enjoys a construction of an iteration operator. Our work complements this approach by focusing on a semantics for interaction, which might be relevant to extend hybrid programs with communication primitives.

Discrete Event Systems Our work represents both cyber and physical aspects of systems in a unified model of discrete event systems [73, 5]. In [56], the author lists the current challenges in modelling cyber-physical systems in such a way. The author points to the problem of modular control, where even though two modules run without problems in isolation, the same two modules may block when they are used in conjunction. In [81], the authors present procedures to synthesize supervisors that control a set of interacting processes and, in the case of failure, report a diagnosis. An application for large scale controller synthesis is given in [72]. Our framework allows for experiments on modular control, by adding an agent controller among the set of agents to be controlled. The implementation in Maude enables the search of, for instance, blocking configurations.

Coordination In [73], the author describes infinite behaviors of process and their synchronization. Notably, the problem of non-blockingness is stated: if two processes eventually interact on some actions, how to make sure that both processes will not block each others. The concept of centrality of a process is introduced.

Transactional components Transactional components such has in Reo have been extensively studied from a formal perspective. In [47], Jongmans presents over 30 semantics for Reo. Current work adds an intermediate linear semantics for Reo, for which proving the correctness of an implementation as given in [77, 78, 45, 32] would be possible. We also believe that the results presented in this paper may be of benefit to other similar semantics, such as [21].

From a trace theoretical perspective, the notion of transaction has also been studied. In [43], elements in $\mathcal{P}_f(\Sigma)^*$ are also called step sequences. However, the axiom $C = DE$ mentioned in [43], for $C, D, E \subseteq \Sigma$ with $D \cap E = \emptyset$, allows one to split a step into two consecutive substeps, which does not apply in our case, because our transactions are atomic and cannot be split. So, we consider the case of traces, where actions are sets. In [37], Gastin investigates the problem of reconstructing sequences of transactions from a sequence of actions investigated. The Foata normal form, defined in [37], Definition 2.10, partitions a trace into sets of mutually independent actions, and is used for this purpose. Unfortunately, not every sequence of transactions emerges as a Foata normal form, which makes the normal form not suitable for our purpose.

Linearization The linearization of transactional behaviors to linear behaviors has been approached in the context of databases queries and transactions. In [35], the

authors consider the problem of serialization, which aims at reordering a sequence of events into a sequence of individual transactions. In their work, a transaction is a sequence of events that starts with a unique beginning symbol and ends with a unique final symbol. Our work relaxes the assumption that each transaction needs two delimiters.

Synchronous and asynchronous The composition of synchronous systems with asynchronous systems has been investigated in the context of interconnecting machines on a network. Globally asynchronous locally synchronous (GALS) is an architecture for designing electronic circuits which addresses the problem of safe and reliable data transfer between independent clock domains [26]. In our paper, we do the opposite: locally asynchronous globally synchronous (LAGS). For example, in the Sync channel, we split the locally synchronous $\{a, b\}$ transaction into asynchronous a and b steps, and recover the $\{a, b\}$ transaction via global synchronization.