Runtime Composition Of Systems of Interacting Cyber-Physical Components

Benjamin Lion[1], Farhad Arbab[1,2], Carolyn Talcott[3]

[1] CWI, Amsterdam, The Netherlands
lion@cwi.nl
[2] Leiden University, Leiden, The Netherlands
arbab@cwi.nl
[3] SRI International, CA, USA
carolyn.talcott@gmail.com

Abstract. The description of concurrent systems as a network of interacting processes helps to reduce the complexity of the specification. The same principle applies for the description of cyber-physical systems as a network of interacting components. We introduce a transition system based specification of cyber-physical components whose semantics is compositional with respect to a family of algebraic products. We give sufficient conditions for execution of a product of cyber-physical components to be correctly implemented by a lazy runtime expansion of the product construction. Our transition system algebra is implemented in the Maude rewriting logic system. As an example, we show that, under a coordination protocol, a set of autonomous energy-aware robots can self-sort themselves on a shared physical grid.

1 Introduction

Cyber-physical systems are highly interactive. Self driving cars are instances of cyber-physical systems with a significant amount of interaction between cyber and physical aspects. The controller in the car periodically samples its environment through its cameras and other sensors, and performs actions to drive the car. Dually, the environment responds to the action of the car by applying the corresponding power on the wheel, consumming energy, and eventually moving the car on the ground. The specification of a problem involving parts with cyber-physical aspects is complex and requires a specification of each individual part, plus how the parts interact. For instance, consider a car rental agency, for which autonomous cars are parked in a line. Having cars parked too far from the agency wastes time for the renters. The agency may therefore want to sort the cars at the end of the day, so that the reserved cars are first in line for the next day. As one can imagine, such a problem involves several parts in interaction. We give hereafter a specification of a simplified version of this problem, that involves sorting robots on a 2 by n grid.

Interaction in Cyber-Physical systems. We simplify the example of self driving car on a rental parking with a set of robots moving on a field. Consider a set of 5 robots, roaming on a grid of size 2 by 5, as displayed in Figure 1. Robots are identified with a unique identifier, and are initially positioned as shown in the left configuration of Figure 1. Each robot is equipped with a battery from which

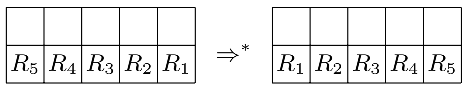

Fig. 1. Initial configuration of the unsorted robots (left), and final configuration of the sorted robots (right).

it draws some energy for its move. A robot can move on an adjacent cell as long as the cell is free, i.e., no other robot is located on the cell. A robot may have a sensor that tells whether the next cell is free, and may send or receive messages from other robots. However, the system of robots is inherently concurrent, as each robot runs at its own speed, draws current from its battery, may sample the environment at arbitrary times, and take decisions according to its own strategy. In the system depicted in Figure 1, can the robots sort themselves in ascending order while maintaining the energy level of their batteries above zero? To answer this question requires analysis of the interactions among both cyber and physical parts of the system.

Specification. The sorting problem highlights the need for a component-based approach to design cyber-physical systems. Both cyber aspects (logic of each robot) and physical parts (grid and batteries of robots) have a decisive contribution for having the robots eventually sorted. Yet, the resulting cyber-physical system is modular: the same set of robots may run on a different grid, with different batteries; or the same grid may welcome different types of robots, with other kinds of sensors.

In [16], we define an algebraic model in which components are first class entities and denote sequences of observables, called Timed-Event Sequences (TES), from both cyber and physical aspects of systems. Interaction between components is defined exogenously using algebraic operators on components. The model of components is declarative: a component denotes a set of sequences of observations, and abstracts from the processes that generate such sequences. A product of two components declares what set of sequences of each component is conserved to comprise the resulting product component, and our algebraic framework supports an open-ended set of product operators parameterized on user-defined composability relations.

In this paper, we give a state-based specification for components to operationally define their behavior. A procedure is then required to generate the behavior of their composition such that the result faithfully respects the interaction constraints among components.

Compositional runtime. The procedure that composes state-based components is either done statically, or dynamically at runtime. In the first case, the resulting composition may be optimized to improve its execution, while the second case allows for modularity and runtime modifications.

Traditionally, a composition is flattened [17] by syntactically enumerating all combinations of states and transitions. The flattened result contains all valid behaviors and therefore faithfully respects the interaction constraints. As state space may quickly get large, flattening the composition may be undesirable.

Instead, we seek a runtime composition operator that jointly executes step-wise each component. The proof obligation for the correctness of such runtime procedure is that the resulting behavior correctly respects the interaction constraints reflected in the product operator over components. For instance, given that component behavior is non-blocking, the runtime should not generate a finite sequence of composite observables for which there is no continuation (non-blockingness [10]). We characterize a set of components for which our step-wise composition is correct: components should be deadlock free and pairwise compatible. As a result, given compatible components, correctness of the step-wise product reduces to showing that after each step, the system of components remains deadlock free. We use our result to analyze in Maude a system of robots that sort themselves.

Contributions. The contributions of this paper are:

– a large family of operators to model interaction of state-based descriptions of cyber-physical components;
– a proof of semantic correctness for a range of user-defined products of TES transition systems;
– a sufficient condition for applying a decomposition operator in incremental steps at runtime;
– an application of our model on an example of self-sorting robots.

The state based model in our formalism allows for a uniform description of arbitrary composition and arbitrary nesting of cyber-physical aspects of components. Such diversity of operators is desirable to model the diversity of interaction among cyber-physical components.

2 Related work

Process algebra The algebra of components described in this paper is an extension of [16]. Algebra of communicating processes [9] (ACP) achieves similar objectives by decoupling processes from their interaction. For instance, the encapsulation operator in process algebra is a unary operator that restricts which actions occurs, i.e., $\delta_H(t \parallel s)$ prevents t and s from performing 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 [5]. In [3], the authors present an extension of ACP to include time-sensitive processes. Our work accommodates the counterparts of the δ_H and γ operators from ACP and provides many more operators needed for direct expression of interaction of cyber-physical components.

Discrete Event Systems In [13], 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 [18], the authors present procedures to synthesize supervisors that control a set of interacting processes and, in the case of failure, report a diagnosis. Cyber-physical systems have also been studied from an actor-model perspective, where actors interact through events [19, 11]. In our work, we add to the event structure a timing constraint, and expose conditions to take the product of discrete event systems at runtime.

Components In [2], the authors give a co-inductive definition of components, to which [16] is an extension. In [4], the authors propose a state based specification as constraint automata. A transition in a constraint automaton is labelled by a guarded command, whose satisfaction depends on the context of its product (other constraint automata). Except from [12], constraint automata do not have time as part of their semantics (i.e., only specify time insensitive components), and only describe observables on ports. In that respect, our model extends constraint automata by generalizing the set of possible observables, and adding the time of the observables as part of the transition.

Timed systems In [8], the authors use heterogeneous timed asynchronous relational nets (HT-ARNs) to model timed sensitive components, and a specification as timed IO-automata. The authors show some conditions (progress-enabledness and r-closure) for the product of two HT-ARNs to preserve progress-enabledness. We may have recovered a similar result, but with some modifications. Our product is more expressive: κ needs not be only synchronization of shared events, but can have more intricate coordination [16] (e.g., exclusion of two events). We do not necessitate our process to be r-closed, and in general, we do not want to explicitly write the silent observations.

The conjunction operator in Timed Automata defines a Timed Automaton whose transitions are either synchronous transition labelled by shared actions (or shared delay), or a transition with an independent action. The conjunction operator, however, is limited and cannot directly express the wide range of relations and compositions that occur within cyber-physical systems. The definition of a parametrized class of operators on TES transition systems makes the interaction constraints explicit in our model and enables modular design of state-based cyber-physical systems.

3 Components in interaction

In [16], we give a unified semantic model to capture cyber and physical aspects of processes as components and characterize their various types of interactions as user-defined products in an algebraic framework. Moreover, we show some general conditions for products on components to be associative, commutative, and idempotent. In this section, we recall the basic definitions of a component and product from [16], and introduce in Section 2.2 some instances of product that suit our example in this paper.

Notations. Given $\sigma : \mathbb{N} \to \Sigma$, let $\sigma[n] \in \Sigma^n$ be the finite prefix of size n of σ and let $\sim_n$ be an equivalence relation on $(\mathbb{N} \to \Sigma) \times (\mathbb{N} \to \Sigma)$ such that $\sigma \sim_n \tau$ if and only if $\sigma[n] = \tau[n]$. Let $FG(L)$ be the set of *left factors* of a set $L \subseteq \Sigma^\omega$, defined as $FG(L) = \{\sigma[n] \mid n \in \mathbb{N},\ \sigma \in L\}$. We use σ' to denote the derivative of the stream σ, such that $\sigma'(i) = \sigma(i+1)$ for all $i \in \mathbb{N}$. We write $\sigma^{(n)}$ for the n-th derivative of σ, i.e., the stream such that $\sigma^{(n)}(i) = \sigma(n+i)$ for all $i \in \mathbb{N}$. 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 $\text{pr}_2(\sigma)(0) \leq \text{pr}_2(\tau)(0)$ (resp. $\text{pr}_2(\tau)(0) \leq \text{pr}_2(\sigma)(0)$) and 0 otherwise.

Let $\mathbb{E}$ be the domain of events. A timed-event stream $\sigma \in \mathit{TES}(E)$ over a set of events $E \subseteq \mathbb{E}$ is an infinite sequence of *observations*, where an observation $\sigma(i) = (O, t)$ consists of a pair of a subset of events in $O \subseteq E$, called *observable*, and a positive real number $t \in \mathbb{R}_+$ as time stamp. A timed-event stream (TES) has the additional properties that consecutive time stamps are increasing and non-Zeno, i.e., for any TES σ and any time $t \in \mathbb{R}$, there exists an element $\sigma(i) = (O_i, t_i)$ in the sequence such that $t < t_i$. For $\sigma \in \mathit{TES}(E)$ 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 $\mathit{dom}(\sigma)$ 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 $t \notin \mathit{dom}(\sigma)$, or there exists an $i \in \mathbb{N}$ such that $\sigma(i) = (\emptyset, t)$. The ambiguity is resolved by checking if $t \in \mathit{dom}(\sigma)$. 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 $\mathit{dom}(\sigma \cup \tau) = \mathit{dom}(\sigma) \cup \mathit{dom}(\tau)$ and $(\sigma \cup \tau)(t) = \sigma(t) \cup \tau(t)$ for all $t \in \mathit{dom}(\sigma) \cup \mathit{dom}(\tau)$.

We recall the greatest post fixed point of a monotone operator, that we later use as a definition scheme and as a proof principle. Let X be any set and let $\mathcal{P}(X) = \{V \mid V \subseteq X\}$ be the set of all its subsets. If $\Psi : \mathcal{P}(X) \to \mathcal{P}(X)$ is a monotone operator, that is, $R \subseteq S$ implies $\Psi(R) \subseteq \Psi(S)$ for all $R \subseteq X$ and $S \subseteq X$, then Ψ has a greatest fixed point $P = \Psi(P)$ satisfying:

$$P = \bigcup \{R \mid R \subseteq \Psi(R)\}$$

This equality can be used as a proof principle: in order to prove that $R \subseteq P$, for any $R \subseteq X$, it suffices to show that R is a post-fixed point of Ψ, that is, $R \subseteq \Psi(R)$.

3.1 Components

A component uniformly models both cyber and physical aspects through a sequence of observables.

Definition 1 (Component). *A* component $C = (E, L)$ *is a pair of an* interface E*, and a* behavior $L \subseteq \mathit{TES}(E)$*.* $\triangle$

A complex system typically consists of multiple components that interact with each other. 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. Formally, an interaction signature $\Sigma = (R, \oplus)$ is a pair of a composability relation $R(E_1, E_2) \subseteq \mathit{TES}(E_1) \times \mathit{TES}(E_2)$ and a composition function $\oplus : \mathit{TES}(\mathbb{E}) \times \mathit{TES}(\mathbb{E}) \to \mathit{TES}(\mathbb{E})$ for arbitrary sets of events $E_1, E_2 \subseteq \mathbb{E}$. 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.

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 *TES*s in the behavior of its operands deemed composable by the composability relation.

Definition 2 (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*

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

While the behaviors of a component are streams, it is natural to consider termination of a component. We express a terminating behavior of component $C = (E, L)$ as an element $\sigma \in L$ such that there exists $n \in \mathbb{N}$ with $\sigma^{(n)} \in \mathit{TES}(\emptyset)$. In other words, a terminating behavior σ is such that, starting from the n-th observation, all next observations are empty.

Given a component C, we define C^* to be the component that may terminate after every sequence of observables. Formally, C^* is the component whose behavior is the prefix closure of C, i.e., the component $C^* = (E, L^*)$, where

$$L^* = L \cup \{\tau \mid \exists n \in \mathbb{N}. \exists \sigma \in L.\ \tau \sim_n \sigma,\ \tau^{(n)} \in \mathit{TES}(\emptyset)\}$$

In [16], we give a co-inductive definition for some R and $\oplus$ given a composability relation on observations, and a composition function on observations.

Let $\kappa(E_1, E_2) \subseteq (\mathcal{P}(E_1) \times \mathbb{R}_+) \times (\mathcal{P}(E_2) \times \mathbb{R}_+)$ be a composability relation on observations, and, for any $\mathcal{R} \subseteq \mathit{TES}(E_1) \times \mathit{TES}(E_2)$, let $\Phi_\kappa(E_1, E_2)(\mathcal{R}) \subseteq \mathit{TES}(E_1) \times \mathit{TES}(E_2)$ 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 *TES*s, 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 pairs $E_1, E_2 \subseteq \mathbb{E}$, i.e., the relation $[\kappa](E_1, E_2) = \bigcup_{\mathcal{R} \subseteq TES(E_1) \times TES(E_2)} \{\mathcal{R} \mid \mathcal{R} \subseteq \Phi_\kappa(E_1, E_2)(\mathcal{R})\}$.

Two observations are synchronous if the two following conditions hold:

1. every observable that shares an event with the other component interface must occur simultaneously with one of its related observables; and
2. only an observable that does not share events with the other component interface can happen before another observable, i.e., at a strictly lower time.

Definition 3 (Synchronous observations). *We define κ^{sync} as the* synchronous *composability relation on observations and $((O_1, t_1), (O_2, t_2)) \in \kappa^{sync}(E_1, E_2)$ 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 $O_1 \cap E_2 = O_2 \cap E_1$;*

Let $\bowtie$ be the product defined as $\bowtie = \times_{([\kappa^{sync}], \cup)}$. Intuitively, $\bowtie$ synchronizes all observations that contain events shared by the interface of two components. As a result of [16], $\bowtie$ is associative and commutative. Section 3.2 introduces a motivating example in which robots, roaming on a shared physical medium, must coordinate to sort themselves. We define algebraically the system consisting of 5 robots and a grid, to which we then add some coordinating protocol components. For more details on each component, see [14].

3.2 Self-sorting robots

We consider the battery, robot, and grid components introduced in [14] in the following interaction:

$$Sys(n, T_1, \ldots, T_n) = \otimes_{i \in \{1,\ldots,n\}} (R(i, T_i) \times_{\Sigma_{R_i B_i}} B_i) \times_{\Sigma_{RG}} G_\mu(\{1, \ldots, n\}, n, 2)$$

where n is the number of robots $R(i, T_i)$, each interacting with a private battery B_i under the interaction signatures $\Sigma_{R_i B_i}$, and in product with a grid G under the interaction signature Σ_{RG}. We use $\otimes$ for the product with the free interaction signature (i.e., every pair of TESs is composable), and the notation $\otimes_{i \in \{1,\ldots,n\}} \{C_i\}$ for $C_1 \otimes \ldots \otimes C_n$ as $\otimes$ is commutative and associative.

We fix $n = 5$ and the same period T for each robot. We write E for the set of events of the composite system $Sys(5, T)$, and R_i for robot $R(i, T)$ with identifier i. Figure 1 in the introduction shows five robot instances, each of which has a unique and distinct natural number assigned, positioned at an initial location on a grid. The goal of the robots in this example is to move around on the grid such that they end up in a final state where they line-up in the sorted order according to their assigned numbers.

We consider trace properties $P \subseteq TES(E)$ and say that C satisfies P if and only if $L \subseteq P$, i.e., all the behavior of C is included in the property P. For the system $Sys(5, T)$, we consider the following property: *eventually, the* *position of each robot R_i is* $(i,0)_{R_i}$, i.e., every robot successfully reaches its place. This property is a trace property, which we call P_{sorted} and consists of every behavior $\sigma \in TES(E)$ such that there exists an $n \in \mathbb{N}$ with $\sigma(n) = (O_n, t_n)$ and $(i,0)_{R_i} \in O_n$ for all robots R_i. In Section 3.3, we explore ways to enforce the property P_{sorted} on the system of robots, and in Section 5 verify its validity given an operational specification for each robot given in Section 4.

3.3 Properties of components and coordination

Robots may beforehand decide on some strategies to swap and move on the grid such that their composition satisfies the property P_{sorted}. For instance, consider the following strategy for each robot R_i:

– *swapping*: if the last read (x,y) of its location is such that $x < n$, then moves North, then West, then South.
– *pursuing*: otherwise, move East.

Remember that the grid prevents two robots from moving to the same cell, which is therefore removed from the observable behavior. We emphasize that some sequences of moves for each robot may deadlock, and therefore are not part of the component behavior of the system of robots, but may occur operationally by constructing a behavior step-by-step (see Section 4.2). Consider Figure 2, for which each robot follows its internal strategy. Because of non-determinism introduced by the timing of each observations, one may consider the following sequence of observations: first, R_1 moves North, then West; in the meantime, R_2 moves West, followed by R_3, R_4, and R_5. By a similar sequence of moves, the set of robots ends in the configuration on the right of Figure 2. In this position and for each robot, the next move dictated by its internal strategy is disallowed, which corresponds to a *deadlock*. While behaviors do not contain finite sequences of observations, which makes the scenario of Figure 2 not expressible as a TES, such scenario may occur in practice. We give in next Section some analysis to prevent such behavior to happen.

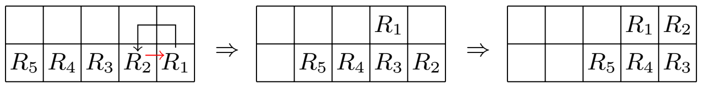

Fig. 2. Initial state of the unsorted robot (left) leading to a possible deadlock (right) if each robot follows its strategy.

Alternatively, the collection of robots may be coordinated by an external protocol that guides their moves. Besides considering the robot and the grid components, we add a third kind of component that acts as a coordinator. In other words, we make the protocol used by the robots to interact explicit and external to them and the grid; i.e., we assume exogenous coordination. Exogenous coordination allows robots to decide a priori on some strategies to swap and move on the grid, in which case their external coordinator component merely unconditionally facilitates their interactions. Alternatively, the external coordinator component may implement a protocol that guides the moves of a set of clueless robots into their destined final locations. The most intuitive of such coordinators is the property itself as a component. Indeed, let $C_{sorted} = (E, L)$ be such that $E = \bigcup_{i \in I} E_{R_i}$ with $I = \{1, 2, 3, 4, 5\}$ and $L = P_{sorted}$. Then, and as shown in [16], the coordinated component

$$Sys(5, T) \bowtie C_{sorted}$$

trivially satisfies the property P_{sorted}. While easily specified, such coordination component is non-deterministic and not easily implementable. We provide an example of a deterministic coordinators.

As discussed, we want to implement the property P_{sorted} as a collection of small coordinators that swap the position of unsorted robots. Intuitively, this protocol mimics the behavior of bubble sort, but for physical devices. Given two robot identifiers R_1 and R_2, we introduce the swap component $S(R_1, R_2)$ that coordinates the two robots R_1 and R_2 to swap their positions. Its interface $E_S(R_1, R_2)$ contains the following events:

- $\mathit{start}(S(R_1, R_2))$ and $\mathit{end}(S(R_1, R_2))$ that respectively notify the beginning and the end of an interaction with R_1 and R_2. Those events are observed when the swap protocol is starting or ending an interaction with either R_1 or with R_2.
- $(x, y)_{R_1}$ and $(x, y)_{R_2}$ that occur when the protocol reads, respectively, the position of robot R_1 and robot R_2,
- d_{R_1} and d_{R_2} for all $d \in \{N, W, E, S\}$ that occur when the robots R_1 and R_2 move;
- $\mathit{lock}(S(R_1, R_2))$ and $\mathit{unlock}(S(R_1, R_2))$ that occur, respectively, when another protocol begins and ends an interaction with either R_1 and R_2.

The behavior of a swapping protocol $S(R_1, R_2)$ is such that it starts its protocol sequence by an observable $\mathit{start}(S(R_1, R_2))$, then it moves R_1 North, then R_2 East, then R_1 West and South. The protocol starts the sequence only if it reads a position for R_1 and R_2 such that R_1 is on the cell next to R_2 on the x-axis. Once the sequence of moves is complete, the protocol outputs the observable end(S(R_1, R_2)). If the protocol is not swapping two robots, or is not locked, then robots can freely read their positions.

Swapping protocols interact with each others by locking other protocols that share the same robot identifiers. Therefore, if S(R_1,R_2) starts its protocol sequence, then S(R_2, R_i) synchronizes with a locked event $\mathit{lock}(S(R_2, R_i))$, for $2 < i$. Then, R_2 cannot swap with other robots unless S(R_1,R_2) completes its sequence, in which case $\mathit{end}(S(R_1, R_2))$ synchronizes with $\mathit{unlock}(S(R_2, R_i))$ for $2 < i$. We extend the underlying composability relation κ on observations such that, for $i < j$, simultaneous observations (O_1, t) and (O_2, t) are composable, i.e., $((O_1, t), (O_2, t)) \in \kappa$, if:

$$\begin{aligned} \mathit{start}(S(R_i, R_j)) \in O_1 \implies & \exists k. k < i. \mathit{lock}(S(R_k, R_i)) \in O_2 \vee \\ & \exists k. j < k. \mathit{lock}(S(R_j, R_k)) \in O_2 \end{aligned}$$

and

$$\begin{aligned} \mathit{end}(S(R_i, R_j)) \in O_1 \implies & \exists k < i. \mathit{unlock}(S(R_k, R_i)) \in O_2 \vee \\ & \exists j < k. \mathit{unlock}(S(R_j, R_k)) \in O_2 \end{aligned}$$

For each pair of robots R_i, R_j such that $i < j$, we introduce a swapping protocol S(R_i, R_j). As a result, the coordinated system is given by the following:

$$\mathit{Sys}(5, T) \bowtie_{i<j} S(R_i, R_j)$$

Note that the definition of $\bowtie$ imposes that, if one protocol starts its sequence, then all protocols that share some robot identifiers synchronize with a lock event. Similar behavior occurs at the end of the sequence. See example 2 for an operational specification of the robot, grid, and swap component.

4 An operational specification of components

In Section 3.1, we give a declarative specification of components, and consider infinite behaviors only. We give, in Section 4.1, an operational specification of components using TES transition systems. We relate the parametrized product of TES transition systems with the parametrized product on their corresponding components, and show its correctness. The composition of two TES transition systems may lead to transitions that are not composable, and ultimately to a deadlock, i.e., a state with no outgoing transitions.

4.1 TES transition systems.

The behavior of a component as in Definition 1 is a set of TESs. We give an operational definition of such set using a labelled transition system.

Definition 4 (TES transition system). *A TES transition system is a triple* $(Q, E, \rightarrow)$ *where* Q *is a set of state identifiers,* E *is a set of events, and* $\rightarrow \subseteq (Q \times \mathbb{N}) \times (\mathcal{P}(E) \times \mathbb{R}_+) \times (Q \times \mathbb{N})$ *is a labelled transition relation, where labels on transitions are observations and a state is a pair of a state identifier and a counter value, such that* $[q, c] \xrightarrow{(O,t)} [q', c']$ *implies that* $c' \geq c$.

We use the notation $\theta([q, c])$ to refer to the counter value c labeling the state.

Example 1 (Strictly progressing TES transition system). We call a TES transition systems *strictly progressing* if, for all transitions $[q, c] \xrightarrow{(O,t)} [q', c']$, we have that $c' > c$. An example of a TES transition system that is strictly progressing is one for which the counter label increases by 1 for every transition, i.e., $[q, c] \xrightarrow{(O,t)} [q', c + 1]$.

Remark 1. The counter value labeling a state of a TES transition system is related to the number of transitions a TES transition system has taken. The counter value is therefore not related to the time of the observation labeling the transition. However, it is possible for some transitions in the TES transition system to keep the same counter value in the post state. As shown later, we use the counter value to model fairness in the product of two TES transition systems.

We present two different ways to give a semantics to a TES transition system: inductive and co-inductive. Both definitions give the same behavior, as shown in Theorem 1, and we use interchangeably each definition to simplify the proofs of, e.g., Theorem 2.

Semantics 1 (runs). A run of a TES transition system is an infinite sequence of consecutive transitions, such that the sequence of observations labeling the transitions form a TES, and the counter in the state is always eventually strictly increasing. Formally, the set of runs $\mathcal{L}^{\text{inf}}(T, s_0)$ of a TES transition system $T = (Q, E, \rightarrow)$ initially in state s_0 is:

$$\mathcal{L}^{\text{inf}}(T, s_0) = \{\tau \in \mathit{TES}(E) \mid \exists \chi \in (Q \times \mathbb{N})^{\omega}.\chi(0) = s_0 \wedge \forall i.\chi(i) \xrightarrow{\tau(i)} \chi(i+1) \wedge \\ \exists j > 0.\ \theta(\chi(i+j)) > \theta(\chi(i))\}$$

Note that the domain of quantification for $\mathcal{L}^{\text{inf}}(T, s_0)$ ranges over TESs, therefore the time labeling observations is, by definition, strictly increasing and non-Zeno. The component semantics of a TES transition system $T = (Q, E, \rightarrow)$ initially in state q is the component $C = (E, \mathcal{L}^{\text{inf}}(T, q))$.

Semantics 2 (greatest post fixed point) Alternatively, the semantics of a TES transition system is the greatest post fixed point of a function over sets of TESs paired with a state. For a TES transition system $T = (Q, E, \rightarrow)$, let $\mathcal{R} \subseteq \mathit{TES}(E) \times (Q \times \mathbb{N})$. We introduce $\phi_T : \mathcal{P}(\mathit{TES}(E) \times (Q \times \mathbb{N})) \rightarrow \mathcal{P}(\mathit{TES}(E) \times (Q \times \mathbb{N}))$ as the function:

$$\phi_T(\mathcal{R}) = \{(\tau, s) \mid \exists n.\exists p \in (Q \times \mathbb{N}),\ s \xrightarrow{\tau[n]} p\ \wedge \theta(p) > \theta(s) \wedge (\tau^{(n)}, p) \in \mathcal{R}\}$$

where $\tau[n]$ is the prefix of size n of the TES τ.

We can show that ϕ_T is monotone, and therefore ϕ_T has a greatest post fixed point $\Omega_T = \bigcup\{\mathcal{R} \mid \mathcal{R} \subseteq \phi_T(\mathcal{R})\}$. We write $\Omega_T(q) = \{\tau \mid (\tau, s) \in \Omega_T\}$ for any $s \in Q \times \mathbb{N}$. Note that the two semantics coincide.

Theorem 1 (Equivalence). *For all $s \in Q \times \mathbb{N}$, $\mathcal{L}^{\text{inf}}(T, s) = \{\tau \mid (\tau, s) \in \Omega_T\}$.*

The semantics of a TES transition system is defined as the component whose behavior contains all runs of the TES transition system. Operationally, however, the (infinite) step-wise generation of such a sequence does not always return a valid prefix of a run. We introduce finite sequences of observables of a TES transition system, and define a deadlock of a TES transition system as a reachable state without an outgoing transition.

Let $T = (Q, E, \rightarrow)$ be a TES transition system. We write $q \xrightarrow{u} p$ for the sequence of transitions $q \xrightarrow{u(0)} q_1 \xrightarrow{u(1)} q_2 \dots \xrightarrow{u(n-1)} p$, where $u = \langle u(0), \dots, u(n-1)\rangle \in (\mathcal{P}(E) \times \mathbb{R}_+)^n$. We write $|u|$ for the size of the sequence u. We use $\mathcal{L}^{\text{fin}}(T, q)$ to denote the set of finite sequences of observables labeling a finite path in T starting from state q, such that

$$\mathcal{L}^{\text{fin}}(T, s) = \{u \mid \exists p.s \xrightarrow{u} p \wedge \forall i < |u| - 1.u(i) = (O_i, t_i) \wedge t_i < t_{i+1}\}$$

Let $FG(L)$ be the set of left factors of a set $L \subseteq \Sigma^\omega$, defined as $FG(L) = \{\sigma[n] \mid n \in \mathbb{N}, \sigma \in L\}$. We write $\sigma(n)$ for the n-th derivative of σ, i.e., the stream such that $\sigma(n)(i) = \sigma(n + i)$ for all $i \in \mathbb{N}$.

Remark 2 (Deadlock). Observe that $FG(\mathcal{L}^{\text{inf}}(T, q)) \subseteq \mathcal{L}^{\text{fin}}(T, q)$ which, in the case of strict inclusion, captures the fact that some states may have no outgoing transitions and therefore deadlock.

Remark 3 (Abstraction). There may be two different TES transition systems T_1 and T_2 such that $\mathcal{L}^{\text{inf}}(T_1) = \mathcal{L}^{\text{inf}}(T_2)$, i.e., a set of TESs is not uniquely characterized by a TES transition system. In that sense, the TES representation of behaviors is more abstract than TES transition systems.

We use the transition rule $q \xrightarrow{(O,t)} q'$ where the counter is not written to denote the set of transitions

$$[q, c] \xrightarrow{(O,t)} [q', c']$$

for $c' \geq c$ with $c, c' \in \mathbb{N}$.

Example 2. The behavior of a robot introduced earlier is a TES transition system $T_R = (\{q_0\}, E_R, \rightarrow)$ where $q_0 \xrightarrow{(\{e\},t)} q_0$ for arbitrary t in $\mathbb{R}_+$ and $e \in E_R$. Similarly, the behavior of a grid is a TES transition system $T_G(I, n, m) = (Q_G, E_G(I, n, m), \rightarrow)$ where:

- $Q_G \subseteq (I \rightarrow ([0; n] \times [0; m]))$,
- $f \xrightarrow{(O,t)} f'$ for arbitrary t in $\mathbb{R}_+$, such that
  - $d_R \in O$ implies $f'(R)$ is updated according to the direction d if the resulting position is within the bounds of the grid;
  - $(x, y)_R \in O$ implies $f(R) = (x, y)_R$ and $f'(R) = f(R)$;
  - $f'(R) = f(R)$, otherwise.

The behavior of a swap protocol S(R_i,R_j) with $i < j$ is a TES transition system $T_S(R_1, R_2) = (Q, E, \rightarrow)$ where, for $t_1, t_2, t_3 \in \mathbb{R}_+$ with $t_1 < t_2 < t_3$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$;
- $E = E_{R_i} \cup E_{R_j} \cup \{\mathit{lock}(R_i, R_j), \mathit{unlock}(R_i, R_j), \mathit{start}(R_i, R_j), \mathit{end}(R_i, R_j)\}$
- $q_1 \xrightarrow{(\{\mathit{lock}(R_i,R_j)\},t_1)} q_2$;
- $q_2 \xrightarrow{(\{\mathit{unlock}(R_i,R_j)\},t_1)} q_1$;

- $q_1 \xrightarrow{(\{start(R_i,R_j),(x,y)_{R_i},(x+1,y)_{R_j}\},t_1)} q_3$;
- $q_3 \xrightarrow{(\{N_{R_j}\},t_1)} q_4 \xrightarrow{(\{W_{R_j},E_{R_i}\},t_2)} q_5 \xrightarrow{(\{S_{R_j}\},t_3)} q_6$;
- $q_6 \xrightarrow{(\{end(R_i,R_j)\},t_1)} q_1$;

We use the letters E, W, S, and N, for an observation of a robot moving in the directions East, West, South, and North, respectively. ■

The product of two components is parametrized by a composability relation κ on observations and syntactically constructs the product of two TES transition systems.

Definition 5 (Product). *The product of two TES transition systems $T_1 = (Q_1, E_1, \rightarrow_1)$ and $T_2 = (Q_2, E_2, \rightarrow_2)$ under the constraint κ is the TES transition system $T_1 \times_\kappa T_2 = (Q_1 \times Q_2, E_1 \cup E_2, \rightarrow)$ such that:*

$$\frac{[q_i, c_i] \xrightarrow{(O_i,t_i)}_i [q_i', c_i'] \quad i \in \{1,2\} \quad ((O_1,t_1),(\emptyset,t_1)) \in \kappa(E_1,E_2) \quad t_1 < t_2}{[(q_1,q_2), \min(c_1,c_2)] \xrightarrow{(O_1,t_1)} [(q_1',q_2), \min(c_1',c_2)]}$$

$$\frac{[q_i, c_i] \xrightarrow{(O_i,t_i)}_i [q_i', c_i'] \quad i \in \{1,2\} \quad ((\emptyset,t_2),(O_2,t_2)) \in \kappa(E_1,E_2) \quad t_2 < t_1}{[(q_1,q_2), \min(c_1,c_2)] \xrightarrow{(O_2,t_2)} [(q_1',q_2), \min(c_1,c_2')]}$$

$$\frac{[q_i, c_i] \xrightarrow{(O_i,t_i)}_i [q_i', c_i'] \quad i \in \{1,2\} \quad ((O_1,t_1),(O_2,t_2)) \in \kappa(E_1,E_2) \quad t_2 = t_1}{[(q_1,q_2), \min(c_1,c_2)] \xrightarrow{(O_1 \cup O_2,t_1)} [(q_1',q_2'), \min(c_1',c_2')]}$$

Remark 4. The notion of an observation is an abstraction that groups an atomic set of events within an ϵ neighborhood of a time t (see [16]). The statement that two observations happen *at the same time* therefore becomes meaningful, and describes two sets of events that occur atomically within an ϵ neighborhood of the same time.

Observe that the product is defined on pairs of transitions, which implies that if T_1 or T_2 has a state without outgoing transition, then the product has no outgoing transitions from that state. The reciprocal is, however, not true in general. We write $C_{T_1 \times_\kappa T_2}((s_1, s_2))$ for the component $C_{T_1 \times_\kappa T_2}([(q_1,q_2), \min(c_1,c_2)])$ where $s_1 = [q_1, c_1]$ and $s_2 = [q_2, c_2]$.

Theorem 2 states that the product of TES transition systems denotes (given a state) the set of TESs that corresponds to the product of the corresponding components (in their respective states). Then, the product that we define on TES transition systems does not add nor remove behaviors with respect to the product on their respective components.

Example 3. Consider two strictly progressing (as in Example 1) TES transition systems $T_1 = (Q_1, E_1, \rightarrow_1)$ and $T_2 = (Q_2, E_2, \rightarrow_2)$. Then, consider a transition in the product $T_1 \times_\kappa T_2$ such that

$$[(q_1,q_2), c] \xrightarrow{(O_1,t_1)} [(q_1',q_2), c]$$

we can deduce that T_1 made a step while the counter c labelling the state didn't change. Therefore, T_2 in state q_2 has a counter labelling its state that is higher than the counter labelling the state in q_1. Alternatively, if

$$[(q_1, q_2), c] \xrightarrow{(O_1,t_1)} [(q_1', q_2), c+1]$$

then the counter at q_2 may become lower than the counter at which T_1 performs the next transition, which means that eventually T_2 has to take a transition.

The composability relation κ in the product of two TES transition systems (see Definition 5) accepts an independent step from T_1 (resp. T_2) if the observation labeling the step relates to the simultaneous silent observation from T_2 (resp. T_1). Given two composable TESs σ and τ respectively in the component behavior of T_1 and T_2, the composability relation $[\kappa]$ must relate heads of such TESs co-inductively. As we do not enforce silent observations to be effective from the product rules (1) and (2), we consider composability relations such that:

- if $((O_1, t_1), (\emptyset, t_1)) \in \kappa(E_1, E_2)$ then $((O_1, t_1), (O_2, t_2)) \in \kappa(E_1, E_2)$ for any $O_2 \subseteq \mathcal{P}(E_2)$ and $t_2 > t_1$; and
- if $((\emptyset, t_2), (O_2, t_2)) \in \kappa(E_1, E_2)$ then $((O_1, t_1), (O_2, t_2)) \in \kappa(E_1, E_2)$ for any $O_1 \subseteq \mathcal{P}(E_1)$ and $t_1 > t_2$

The two rules above encode that an observation from T_1 is independent to T_2 (i.e., $((O_1, t_1), (\emptyset, t_1)) \in \kappa(E_1, E_2)$ if and only if T_1 and T_2 can make observations at difference times (i.e., $((O_1, t_1), (O_2, t_2)) \in \kappa(E_1, E_2)$ for arbitrary (O_2, t_2) from T_2 with $t_2 > t_1$.

Theorem 2 (Correctness). *For two TES transition systems T_1 and T_2, and for κ satisfying the constraint above:*

$$C_{T_1 \times_\kappa T_2}(s) = C_{T_1}(s_1) \times_{([\kappa],[\cup])} C_{T_2}(s_2)$$

with $s_1 = [q_1, c_1] \in (Q_1 \times \mathbb{N})$, $s_2 = [q_2, c_2] \in (Q_2 \times \mathbb{N})$, *and* $s = [(q_1, q_2), \min(c_1, c_2)]$.

Remark 5 (Fairness). Fairness, in our model, is the property that, in a product of two TESs $T_1 \times_\kappa T_2$, then always, eventually, T_1 and T_2 each makes progress. The definition of the product of two TES transition systems defines the counter value of the composite state as the minimal counter value from the two compound states. The semantic condition that considers runs with counter values that are always eventually increasing is sufficient for having T_1 and T_2 to always eventually take a step, as shown in Theorem 2.

Remark 6. Note that the generality of Theorem 2 comes from the parametrized composability relation κ. Thus, for instance, the synchronous product of I/O automata can be expressed by a suitable composability relation κ that synchronizes the occurrence of shared inputs and outputs for parallel composition or conjunction (see [7]).

We give in Example 4 the TES transition systems resulting from the product of the TES transition systems of two robots and a grid. Example 4 defines operationally the components in Section 2.2, i.e., their behavior is generated by a TES transition system.

Example 4. Let T_{R_1}, T_{R_2} be two TES transition systems for robots R_1 and R_2, and let $T_G(\{1\}, n, m)$ be a grid with robot R_1 alone and $T_G(\{1, 2\}, n, m)$ be a grid with robots R_1 and R_2. We use κ^{sync} as defined in Definition 3.

The product of T_{R_1}, T_{R_2}, and $T_G(\{1, 2\}, n, m)$ under κ^{sync} is the TES transition system $T_{R_1} \times_{\kappa^{sync}} T_{R_2} \times_{\kappa^{sync}} T_G(\{1, 2\}, n, m)$ such that it synchronizes observations of the two robots with the grid, but does not synchronize events of the two robots directly, since the two sets of events are disjoint. ■

As a consequence of Theorem 1, letting κ^{sync} be the composability relation used in the product $\bowtie$ and writing $T = T_{R_1} \times_{\kappa^{sync}} T_{R_2} \times_{\kappa^{sync}} T_G$, $C_T(q_1, q_2, q_3)$ is equal to the component $C_{T_{R_1}}(q_1) \bowtie C_{T_{R_2}}(q_2) \bowtie C_{T_G}(q_3)$.

Definition 6. *Let T be a TES transition system, and let $C_T(q) = (E, \mathcal{L}^{\inf}(T, q))$ be a component whose behavior is defined by T. Then, C is* deadlock free *if and only if $FG(\mathcal{L}^{\inf}(T, q)) = \mathcal{L}^{\text{fin}}(T, q) \neq \emptyset$. As a consequence, we also say that (T, q) is deadlock free when $C_T(q)$ is deadlock free.*

A class of deadlock free components is that of components that accept arbitrary insertions of $\emptyset$ observables in between two observations. We say that such component is *prefix-closed*, as every sequence of finite observations can be continued by an infinite sequence of empty observables, i.e., C is such that $C = C^*$ (as defined after Definition 5). We say that a TES transition system T is prefix-closed in state s if and only if $C_T(s) = C^*_T(s)$. For instance, if T is such that, for any state s and for any $t \in \mathbb{R}_+$ there is a transition $s \xrightarrow{(\emptyset, t)} s$, then T is prefix-closed.

Lemma 1. *If T_1 and T_2 are prefix-closed in s_1 and s_2 respectively, then $T_1 \times_{\kappa^{sync}} T_2((s_1, s_2))$ is prefix-closed.*

We search for the condition under which deadlock freedom is preserved under a product. Section 3.3 gives a condition for the product of two deadlock free components to be deadlock free.

4.2 Compatibility of TES transition systems

Informally, the condition of κ-compatibility of two TES transition systems T_1 and T_2, respectively in initial state s_{01} and s_{02}, describes the existence of a relation $\mathcal{R}$ on pairs of states of T_1 and T_2 such that:

- $(s_{01}, s_{02}) \in \mathcal{R}$, and
- for every state $(s_1, s_2) \in \mathcal{R}$, there exists an outgoing transition from T_1 (reciprocally T_2) that composes under κ with an outgoing transition of T_2 (respectively T_1) and the resulting pair of states is in the relation $\mathcal{R}$.

Formally, a TES transition system $T_1 = (Q_1, E_1, \rightarrow_1)$ from state s_{01} is κ-compatible with a TES transition system $T_2 = (Q_2, E_2, \rightarrow_2)$ from state s_{02}, and we say (T_1, s_{01}) is κ-compatible with (T_2, s_{02}) if there exists a relation $\mathcal{R} \subseteq (Q_1 \times \mathbb{N}) \times (Q_2 \times \mathbb{N})$ such that $(s_{01}, s_{02}) \in \mathcal{R}$ and for any $(s_1, s_2) \in \mathcal{R}$,

- there exist $s_1 \xrightarrow{(O_1,t_1)}_1 s_1'$ and $s_2 \xrightarrow{(O_2,t_2)}_2 s_2'$ such that $((O_1, t_1), (O_2, t_2)) \in \kappa(E_1, E_2)$; and
- for all $s_1 \xrightarrow{(O_1,t_1)}_1 s_1'$ and $s_2 \xrightarrow{(O_2,t_2)}_2 s_2'$ if $((O_1, t_1), (O_2, t_2)) \in \kappa(E_1, E_2)$ then $(u_1, u_2) \in \mathcal{R}$, where $u_i = s_i$ if $t_i = \min\{t_1, t_2\}$, and $u_i = s_i'$ otherwise for $i \in \{1, 2\}$.

In other words, if (T_1, s_1) is κ-compatible with (T_2, s_2), then there exists a composable pair of transitions in T_1 and T_2 from each pair of states in $\mathcal{R}$ (first item of the definition), and all pairs of transitions in T_1 composable with a transition in T_2 from a state in $\mathcal{R}$ end in a pair of states related by $\mathcal{R}$. If (T_2, s_2) is κ-compatible to (T_1, s_1) as well, then we say that (T_1, s_1) and (T_2, s_2) are κ-compatible.

Theorem 3 (Deadlock free). *Let (T_1, s_1) and (T_2, s_2) be κ-compatible. Let $C_{T_1}(s_1)$ and $C_{T_2}(s_2)$ be deadlock free, as defined in Definition 6. Then, $C_{T_1}(s_1) \times_{([\kappa],[\cup])} C_{T_2}(s_2)$ is deadlock free.*

In general however, κ-compatibility is not preserved over products, as demonstrated by Example 5. For the case of coordinated cyber-physical systems, components are usually not prefix-closed as there might be some timing constraints or some mandatory actions to perform in a bounded time frame.

Example 5. Suppose three TES transition systems $T_i = (\{q_i\}, \{a, b, c, d\}, \rightarrow_i)$, with $i \in \{1, 2, 3\}$, defined as follows for all $n \in \mathbb{N}$:

- $q_1 \xrightarrow{(\{a,b\},n)}_1 q_1$ and $q_1 \xrightarrow{(\{a,c\},n)}_1 q_1$;
- $q_2 \xrightarrow{(\{a,c\},n)}_2 q_2$ and $q_2 \xrightarrow{(\{a,d\},n)}_2 q_2$;
- $q_3 \xrightarrow{(\{a,d\},n)}_3 q_3$ and $q_3 \xrightarrow{(\{a,b\},n)}_3 q_3$.

The TES transition systems $T_1(q_1)$, $T_2(q_2)$, and $T_3(q_3)$ are pairwise κ^{sync}-compatible because each pair-wise product has an outgoing transition with an infinite run. However, $T_1(q_1)$ is not κ^{sync}-compatible with $T_2(q_2) \times_{\kappa^{sync}} T_3(q_3)$ because no transition can synchronize between all three TES transition systems. ■

Lemma 2. *Let $\times_\kappa$ be commutative and associative, and for arbitrary $E_1, E_2 \in \mathbb{E}$, and $t \in \mathbb{R}_+$, let $((\emptyset, t), (\emptyset, t)) \in \kappa(E_1, E_2)$. Moreover, let S be a set of TES transition systems, such that for $T \in S$ and every state $[q, n]$ in T, we have $[q, n] \xrightarrow{(\emptyset,t)} [q, n]$. For $S = S_1 \uplus S_2$ a partition of S, $\times_\kappa\{T\}_{T \in S_1}$ and $\times_\kappa\{T\}_{T \in S_2}$ are κ-compatible and the component $C_{\times_\kappa\{T\}_{T\in S}}$ is deadlock free.*

The consequence of two TES transition systems T_1 and T_2 being κ-compatible on (s_1, s_2) and deadlock free, is that they can be run *step-by-step* from (s_1, s_2)

and ensure that doing so would not generate a sequence of observations that is not a prefix of an infinite run. However, there is still an obligation for the *step-by-step* execution to produce a run that is in the behavior of the product, i.e., to perform a step-by-step product at runtime. Indeed, the resulting sequence of states must always increase the counter value, which means that the selection of a step must be *fair* (as introduced in Remark 5). We show in Example 6 an example for which an infinite sequence of transitions in the product (e.g., produced by a step-by-step implementation of the product) would not yield a run, due to fairness violation.

Example 6. Let $T_1 = (\{q_1\}, \{a\}, \rightarrow_1)$ and $T_2 = (\{q_2\}, \{b\}, \rightarrow_2)$ be two TES transition systems such that: $[q_1, c] \xrightarrow{(\{a\},t)}_1 [q_1, c+1]$ and $[q_2, c] \xrightarrow{(\{b\},t)}_2 [q_2, c+1]$for all $t \in \mathbb{R}_+$ and all $c \in \mathbb{N}$. Let κ be such that $((\{a\},t),(\emptyset,t)) \in \kappa(\{a\},\{b\})$ and $((\emptyset,t),(\{b\},t)) \in \kappa(\{a\},\{b\})$. Then, the product $T_1 \times_\kappa T_2$ has the composite transitions $[(q_1, q_2), c] \xrightarrow{(\{a\},t)} [(q_1, q_2), c]$ and $[(q_1, q_2), c] \xrightarrow{(\{b\},t)} [(q_1, q_2'), c]$ for all $c \in \mathbb{N}$ and $t \in \mathbb{R}_+$.

The product, therefore has runs of the kind $[(q_1, q_2), c] \xrightarrow{(\{a\},t_i)} [(q_1', q_2), c]$ where for all $i \in \mathbb{N}$, $c_i + 1 = c_{i+1}$ and $t_i < t_{i+1}$ (increasing) and there exists $j \in \mathbb{N}$ with $i < t_j$ (non-Zeno). Thus, this run does only transitions from T_1 and none from T_2: there is a step for which the counter c does not increase anymore. One reason is that rule (1) of the product is always chosen. Instead, by imposing that we always eventually take rule (3), we ensure that the step-by-step product is fair.

We consider a class of TES transition systems for which a *step-by-step* implementation of their product is fair, i.e., always eventually the counter of the composite state increases. More particularly, we consider TES transition systems that always eventually require synchronization. Therefore, the product always eventually performs rule (3), and the runs are consequently fair. Such property is a composite property, that can be obtained compositionally by imposing a trace property on a TES transition system, such as: for every trace, there is always eventually a state for which all outgoing transitions must synchronize with an observation from the other TES transition system.

Remark 7. In the actor model, fairness is usually defined as an individual property: always eventually an action that is enabled (such as reading a message in a queue) will be performed. This notion of fairness differs from the one we introduced for TES transition systems. In our model, fairness formalizes a collective property, namely that each component always eventually progresses to yield an observation.

Definition 7 (k-synchronizing). *Two TES transition systems T_1 and T_2 are k-synchronizing under κ if every sequences of k transitions in the product $T_1 \times_\kappa T_2$ contains at least one transition constructed by rule* (3) *of the product in Definition 5.*

Lemma 3. *Let T_1 and T_2 be two k-synchronizing TES transition systems. Then, a step-by-step execution of the product $T_1 \times_\kappa T_2$ is fair, namely, every finite sequence of transitions is a prefix of an infinite run in the product behavior, i.e., $FG(\mathcal{L}^{inf}(T_1 \times_\kappa T_2, q)) = \mathcal{L}^{fin}(T_1 \times_\kappa T_2, q)$.*

Remark 8. The step-by-step implementation of the product is sound if TES transition systems always eventually synchronize on a transition. Definition 7 and Lemma 3 show that if two TES transition systems are k-synchronizing, then their product can be formed lazily, step-by-step, at runtime.

5 Application: self-sorting robots

We implemented in Maude a framework to simulate concurrent executions of TES transition systems, where time stamps are restricted to natural numbers. Using the description given in Example 2 for the grid and for robots, we add to their composition several protocols that aim at preventing deadlock. The source for the implementation is accessible at [1] to reproduce the results of this section.

Components in Maude The implementation of TES transition systems in Maude focuses on a subset that has some properties. First, TES transition systems in Maude have time stamps that range over the set of positive natural numbers $\mathbb{N}$. We do not implement components with real time.

Second, TES transition systems run at a fixed sampling rate. Let T be the sampling period. This property encodes that, between two transitions in the TES transition system, a fixed time duration of T has passed. A TES transition system may allow for arbitrary delay of its transitions by a fixed multiple k of delay T. In which case, we say that the TES transition system is *delay insensitive.* Formally, for every $q \xrightarrow{(O,n)} p$ of a delay insensitive TES transition system with period T, we have $n = k \cdot T$ for some $k \in \mathbb{N}$. We therefore write $q \xrightarrow{O} p$ to denote the set of transitions $q \xrightarrow{(O,k\cdot T)} p$ for all $n \in \mathbb{N}$.

In Maude, the state of a TES transition system component is represented by a term and the state of a composed system is a multiset of component states. Transitions of the step-wise product are defined in terms of such system states. For instance, the swap protocol between robots $R(3)$ and $R(1)$ is the defined in Maude as:

```
[swap(R(3),R(1)): Protocol | k("s") |-> ds(q(0)); false; mt]
```

where `swap(R(3),R(1))` is the name of the component; `Protocol` is its class; `k("s")` maps to the initial state of the protocol `q(0)`; `"false"` denotes the status of the protocol; and `"mt"` is the set of transitions that the protocol may take.

Runtime composition. The product of TES transition systems is constructed at runtime, step by step. We use κ^{sync} for the product of TES transition systems.

Given a list of initialized TES transition system, the runtime computes the set of all possible composite transitions, from which transitions that violate the composability relation κ^{sync} are filtered out, and one transition that is composable is non-deterministically chosen.

Algorithm 1 Runtime composition

```
Require:
    - $n$ initialized TES transition systems $S = \{T_1(q_1), \dots, T_n(q_n)\}$
 1: procedure RuntimeComposition
 2:     for $T_i(q_i) \in S$ do add $\{q_i \xrightarrow{O_i}_i p_i \mid p_i \in Q_i\}$ to $Tr$
 3:     while $trs_i, trs_j \in Tr$ do
 4:         for $q_i \xrightarrow{O_i} p_i \in trs_i$ and $q_j \xrightarrow{O_j} p_j \in trs_j$ do
 5:             if $((O_i, 1), (O_j, 2)) \in \kappa^{sync}(E_i, E_j)$ then
 6:                 add $(q_i, q_j) \xrightarrow{O_i} (p_i, q_j)$ to $trs_{ij}$
 7:             if $((O_i, 2), (O_j, 1)) \in \kappa^{sync}(E_i, E_j)$ then
 8:                 add $(q_i, q_j) \xrightarrow{O_j} (q_i, p_j)$ to $trs_{ij}$
 9:             if $((O_i, 1), (O_j, 1)) \in \kappa^{sync}(E_i, E_j)$ then
10:                 add $(q_i, q_j) \xrightarrow{O_i \cup O_j} (p_i, p_j)$ to $trs_{ij}$
11:         $Tr := (Tr \setminus \{trs_i, trs_i\}) \cup \{trs_{ij}\}$
12:     let $trs \in Tr$
13:     let $(q_1, \dots, q_n) \xrightarrow{O} (r_1, \dots, r_n) \in trs$
14:     for $i \leq n$ do $T_i(q_i) \Rightarrow T_i(r_i)$
```

Algorithm 1 shows the procedure RuntimeComposition that corresponds to a one step product of the input TES transition systems. Note that such procedure applied recursively on its results would generate a behavior that is in behavior of the product of the TES transition systems.

Results. Initially, the system consists of three *robots*, with identifiers *R(0)*, *R(1)*, and *R(2)*, each coordinated by two protocols *swap*$(R(i), R(j))$ with $i, j \in \{0, 1, 2\}$ and $j < i$. The trolls move on a grid and trolls *R(0)*, *R(1)*, and *R(2)* are respectively initialized at position $(2; 0)$, $(1; 0)$, and $(0; 0)$.[4] The property P_{sorted} is a reachability property on the state of the grid, that states that *eventually, all robots are in the sorted position.* In Maude, given a system of 3 robots, we express such reachability property with the following search command:

```
search [1] init =>*
    [sys::Sys  [ field : Field | k((0;0)) |-> d(R(0)),
                k((1;0)) |-> d(R(1)),
                k((2;0)) |-> d(R(2)) ; true ; mt]] .
```

The initial configuration of the grid is such that robot 0 is on location $(2; 0)$, robot 1 on $(1; 0)$, and robot 2 on $(0; 0)$. Since the grid is of size 3 by 2, robots need to coordinate to reach the desired sorted configuration. The search commands search for a final state where the robots are sorted.

[4] We refer to [6] for a more detailed description of the Maude framework.

Table 1 features three variations on the sorting problem. The first system is composed of robots whose moves are free on the grid. The second adds one battery for each component, whose energy level decreases for each robot move. The third system adds a swap protocol for every pair of two robots. The last system adds a protocol and batteries to compose with the robots.

We record, for each of those systems, whether the sorted configuration is reachable (P_{sorted}), and if all three robots can run out of energy (P_{bat}).

Table 1. Evaluation of different systems for the P_{sorted} and P_{bat} behavioral properties, where *st.* stands for states, *rw* for rewrites. Note that the P_{bat} property is not evaluated when the system does not contain battery components.

System	P_{sorted}	P_{bat}
$\bowtie_{0\leq i\leq 2} R_i \bowtie G$	12.10^3 st., 25s, 31.10^6 rw	
$\bowtie_{0\leq i\leq 2} (R_i \bowtie B_i) \bowtie G$	12.10^3 st., 25s, 31.10^6 rw	true
$\bowtie_{0\leq i\leq 2} R_i \bowtie G \bowtie_{0\leq i<j\leq 2} S(R_i, R_j)$	8250 st., 44s, 80.10^6 rw	
$\bowtie_{0\leq i\leq 2} (R_i \bowtie B_i) \bowtie G \bowtie_{0\leq i<j\leq 2} S(R_i, R_j)$	8250 st., 71s, 83.10^6 rw	false

Observe that the reachability query returns a solution for both system: the one with and without protocols. However, the time to reach the first solution increases as the number of transition increases (adding the protocol components). We leave as future work some optimizations to improve on our results.

6 Conclusion

We introduce a transition system based specification of cyber-physical systems whose semantics is compositional with respect to a family of algebraic products. We give sufficient conditions for execution of a product to be correctly implemented by a lazy expansion of the product construction. We proved, using an implementation of our framework in Maude, a set of autonomous robots that move on a grid, coordinated by a local swapping protocol, satisfy the emergent property of ending in sorted position.

This work is a first step towards a finite characterization of component behaviors. We give in [15] a specification of TES transition systems as rewriting agents, and explore other case studies for showing safety properties of cyber-physical systems. As a future work, the extension of the framework with real time can open reasoning about optimal frequencies at which robots can interact to fulfill a coordination pattern.

Acknowledgement Talcott was partially supported by the U. S. Office of Naval Research under award numbers N00014-15-1-2202 and N00014-20-1-2644, and

NRL grant N0017317-1-G002. Arbab was partially supported by the U. S. Office of Naval Research under award number N00014-20-1-2644. We thank the reviewers for their critical comments and their helpful suggestions.

References

[1] `https://scm.cwi.nl/CSY/cp-agent`.

[2] Farhad Arbab and Jan Rutten. "A Coinductive Calculus of Component Connectors". In: Sept. 2002, pp. 34–55. ISBN: 978-3-540-20537-1. DOI: `10.1007/978-3-540-40020-2_2`.

[3] Jos C. M. Baeten and Cornelis A. Middelburg. "Real time process algebra with time-dependent conditions". In: *J. Log. Algebraic Methods Program.* 48.1-2 (2001), pp. 1–38. DOI: `10.1016/S1567-8326(01)00004-2`.

[4] Christel Baier et al. "Modeling component connectors in Reo by constraint automata". In: *Science of Computer Programming* 61.2 (2006). Second International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'03), pp. 75–113. ISSN: 0167-6423. DOI: `https://doi.org/10.1016/j.scico.2005.10.008`.

[5] J.A. Bergstra and J.W. Klop. "Process algebra for synchronous communication". In: *Information and Control* 60.1 (1984), pp. 109–137. ISSN: 0019-9958. DOI: `https://doi.org/10.1016/S0019-9958(84)80025-X`.

[6] Manuel Clavel et al. *All About Maude: A High-Performance Logical Framework*. Vol. 4350. LNCS. Springer, 2007.

[7] Alexandre David et al. "Timed I/O automata: a complete specification theory for real-time systems". In: *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, April 12-15, 2010.* Ed. by Karl Henrik Johansson and Wang Yi. ACM, 2010, pp. 91–100. DOI: `10.1145/1755952.1755967`.

[8] José Luiz Fiadeiro and Antónia Lopes. "Heterogeneous and asynchronous networks of timed systems". In: *Theor. Comput. Sci.* 663 (2017), pp. 1–33. DOI: `10.1016/j.tcs.2016.12.014`.

[9] Wan J. Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2000. ISBN: 978-3-540-66579-3. DOI: `10.1007/978-3-662-04293-9`.

[10] A. C. van Hulst, Michel A. Reniers, and Wan J. Fokkink. "Maximally permissive controlled system synthesis for non-determinism and modal logic". In: *Discret. Event Dyn. Syst.* 27.1 (2017), pp. 109–142. DOI: `10.1007/s10626-016-0231-8`.

[11] Tobias Kappé et al. "Soft component automata: Composition, compilation, logic, and verification". In: *Sci. Comput. Program.* 183 (2019). DOI: `10.1016/j.scico.2019.08.001`.

[12] Natallia Kokash, Mohammad Mahdi Jaghoori, and Farhad Arbab. "From Timed Reo Networks to Networks of Timed Automata". In: *Electronic Notes in Theoretical Computer Science* 295 (2013). Proceedings the 9th International Workshop on Formal Engineering approaches to Software

Components and Architectures (FESCA), pp. 11–29. ISSN: 1571-0661. DOI: `https://doi.org/10.1016/j.entcs.2013.04.004`.

[13] Stéphane Lafortune. "Discrete Event Systems: Modeling, Observation, and Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019), pp. 141–159. DOI: `10.1146/annurev-control-053018-023659`.

[14] Benjamin Lion, Farhad Arbab, and Carolyn Talcott. *Runtime Composition Of Systems of Interacting Cyber-Physical Components.* 2022.

[15] Benjamin Lion, Farhad Arbab, and Carolyn L. Talcott. "A Rewriting Framework for Interacting Cyber-Physical Agents". In: *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning - 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22-30, 2022, Proceedings, Part III.* Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 13703. Lecture Notes in Computer Science. Springer, 2022, pp. 356–372. DOI: `10.1007/978-3-031-19759-8\_22`.

[16] Benjamin Lion, Farhad Arbab, and Carolyn L. Talcott. "A Semantic Model for Interacting Cyber-Physical Systems". In: *Proceedings 14th Interaction and Concurrency Experience, ICE 2021, Online, 18th June 2021.* Ed. by Julien Lange et al. Vol. 347. EPTCS. 2021, pp. 77–95. DOI: `10.4204/EPTCS.347.5`.

[17] Sahar Mohajerani, Robi Malik, and Martin Fabian. "A framework for compositional nonblocking verification of extended finite-state machines". In: *Discret. Event Dyn. Syst.* 26.1 (2016), pp. 33–84. DOI: `10.1007/s10626-015-0217-y`.

[18] Meera Sampath, Stéphane Lafortune, and Demosthenis Teneketzis. "Active diagnosis of discrete-event systems". In: *IEEE Trans. Autom. Control.* 43.7 (1998), pp. 908–929. DOI: `10.1109/9.701089`.

[19] Carolyn Talcott. "Cyber-Physical Systems and Events". In: *Software-Intensive Systems and New Computing Paradigms: Challenges and Visions.* Ed. by Martin Wirsing et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 101–115. ISBN: 978-3-540-89437-7. DOI: `10.1007/978-3-540-89437-7_6`.