Heterogeneous Reactive Systems Modeling: Capturing Causality and the Correctness of Loosely Time-Triggered Architectures (LTTA) [*]

Albert Benveniste[†] Benoît Caillaud Luca P. Carloni Paul Caspi
Alberto L. Sangiovanni-Vincentelli

ABSTRACT

We present an extension of a mathematical framework proposed by the authors to deal with the composition of heterogeneous reactive systems. Our extended framework encompasses diverse models of computation and communication such as synchronous, asynchronous, causality-based partial orders, and earliest execution times. We introduce an algebra of tag structures and morphisms between tag sets to define heterogeneous parallel composition formally and we use a result on pullbacks from category theory to handle properly the case of systems derived by composing many heterogeneous components. The extended framework allows us to establish theorems, from which design techniques for correct-by-construction deployment of abstract specifications can be derived. We illustrate this by providing a complete formal support for correct-by-construction distributed deployment of a synchronous design specification over an LTTA medium.

Categories and Subject Descriptors: C.3.3 [Special-purpose and application-based systems]: Real-time and embedded systems.

General Terms: Theory.

Keywords: Heterogeneous reactive systems, distributed deployment, GALS.

[*]This research was supported in part by the European Commission under the projects COLUMBUS, IST-2002-38314, and ARTIST, IST-2001-34820, by the NSF under the project ITR (CCR-0225610), and by the GSRC.

[†]A. Benveniste and B. Caillaud are with Irisa/Inria, Campus de Beaulieu, 35042 Rennes cedex, France; Email: {Albert.Benveniste,Benoit.Caillaud}@irisa.fr; Web pages: www.irisa.fr/sigma2/benveniste/ and www.irisa.fr/prive/Benoit.Caillaud/. L. Carloni and A. Sangiovanni-Vincentelli are with the EECS Department of U.C. Berkeley, Berkeley, CA 94720; Email: {lcarloni,alberto}@eecs.berkeley.edu; Web: www-cad.eecs.berkeley.edu/∼{lcarloni,alberto}; P. Caspi is with Verimag, Centre Equation, 2, rue de Vignate, F-38610 Gieres, Email: caspi@imag.fr; Web: www-verimag.imag.fr/∼caspi.

1. INTRODUCTION

Heterogeneity is a typical characteristic of embedded systems: it naturally manifests itself at the component level where different models of computation may be used to represent component operation and, more frequently, at different levels of abstraction, where, for example, a synchronous-language specification of the design may be refined into a globally asynchronous locally synchronous (GALS) architecture. Having a mathematical framework for the heterogeneous modeling of reactive systems gives freedom of choice between different synchronization policies at different stages of the design process and provides a solid foundation to handle formally communication and coordination among heterogeneous components. Interesting work along similar lines has been the Ptolemy project [9, 12], the MoBIES project [1], the Model-Integrated Computing (MIC) framework [13], and *Interface Theories* [10].

In [5] we introduced *Tagged Systems* as a mathematical model for composing heterogeneous reactive systems. This model is a special case of the Tagged-Signal Model proposed by Lee and Sangiovanni-Vincentelli's [15] (simply referred in the sequel as the LSV model). Common to both approaches is the concept of tags. Tags are used to mark the events of the signals of a system for various purposes. These include indexing the evolution in time (time stamping) of a signal and expressing relations among signal events such as coordination in time (synchrony, asynchrony) and causal dependency. The LSV model is not itself a model of computation or a "grand unified model" but rather a denotational framework to analyze and compare important properties of various models of computations. In the LSV model a system is specified as a set of behaviours, which are sets of events. Each event is associated to a signal and characterized by a data value and a tag.

In developing tagged systems we have built on top of the LSV model. Since we focused on providing a mathematical tool to guide the design of reactive and real-time embedded systems through the assembly of heterogeneous components, we restricted the generality of the LSV model to a particular subset. This subset represents a powerful tool to analyze the design process while it progresses from the system specification to the final implementation through a set of refinement steps. Using tags we can easily express heterogeneous models of computation. Components of different nature present different tag sets and we can formalize their composition by resolving tags and values of the interface signals (unification process). This allows us to capture communication and coordination policies among the components at various levels of abstraction from specification to implementation. In particular, we can formalize the conditions under which the implementation of a synchronous design on distributed loosely-synchronized architecture maintains the original behaviour, i.e. the implementation is *semantics-preserving.*

This paper extends our previous work on tagged systems [5] in three directions:

- Originally, we restricted ourselves to a model in which parallel composition is by intersection, i.e. unifiable events of each component must have identical variable, data value, and tag. While this restriction has allowed us to handle semantic-preserving deployment on GALS architectures, it does not cover all cases of interest. For example, causality relations, scheduling constraints, or earliest execution times are not compatible with parallel composition by intersection. Yet, these are all very important aspects to consider when implementing an embedded system. To capture them, we propose here an extension of tagged systems where the unification rule for tags is itself parameterized.

- In [5], we provided a first set of theorems on the preservation of semantics during distributed deployment. In this paper, we further elaborate on this by introducing an *algebra of tag structures* that allows us to formally define heterogeneous systems. By using morphisms in this algebra and a result on pullbacks from category theory we show how to extend our previous results to the case of systems derived by composing multiple (more than two) heterogeneous components.

  Note that this extension has led us to modify even the bases of our model as presented in [5]. The search for uniformity and compositionality obliged us to define tag morphisms that be true morphisms, *i.e.,* keep their properties through composition (which was not the case of the previous proposal). The result is a more uniform and (seemingly) elegant proposition.

- Finally, to show the use of the theory and prove the effectiveness of the companion *correct-by-construction* methodology, we apply the proposed framework to a fully comprehensive study of deploying a synchronous specification on a loosely time triggered architecture (Ltta) [5, 6]. This architecture is an important concept developed in conjunction with the aerospace industry where a precise notion of time and synchronous events cannot be guaranteed on the field due to the distributed nature of the target architecture.

2. TAGGED SYSTEMS

In this section, we first present a summary of the LSV tagged signal model and then we define our tagged systems and their homogeneous parallel composition.

Throughout this paper, $\mathbf{N} = \{1, 2, \dots\}$ denotes the set of positive integers; $\mathbf{N}$ is equipped with its usual total order $\leq$. $X \mapsto Y$ denotes the set of all partial functions from X to Y.

2.1 The Original LSV Model

We assume an underlying partially ordered set $\mathcal{T}$ of *tags,* we denote by $\leq$ the partial oder on $\mathcal{T}$, and we write $\tau < \tau'$ iff $\tau \leq \tau'$ and $\tau \neq \tau'$. Assume also an underlying set $\mathcal{V}$ of variables, with domain D. Elements of $\mathcal{T} \times D$ are called *events,* and subsets of events are called *signals.* Thus the set of all signals is $S = \mathcal{P}(\mathcal{T} \times D)$. Isomorphically, we also have $S = \mathcal{T} \mapsto \mathcal{P}(D)$.

Tagged systems. In the original LSV model, a tagged system is simply a subset of the set $\mathcal{V} \mapsto S$ of all behaviours, i.e., functions mapping variables to signals.

Parallel composition. In the original LSV model, parallel composition is defined by intersection for tagged systems having identical tag sets.

Discussion. Tags are used to capture relations between events. If the tag set is the set of real numbers, then we can think of this set as time and the induced ordering represents the sequence of events in a signal each with its time of occurrence. If the tag set is the set of natural numbers, then the induced ordering represents "logical" time, i.e., the sequencing of events with no notion of "when" the events occur. This generality allows representing different models of computation and describe their relationship. Of particular interest is the use of tags in systems that are described by a set of signals. In that case, relations among tags of different signals are useful to represent causality relationships that are more complex than the simple causality that occurs when we think of a single signal.

The flexibility of a tag signal model is ideal to represent all phases of a design from specification to implementation. In this case we can march through the various levels of abstraction and understand the refinement of tag sets as going from an "untimed" specification, where tags are minimally constrained, to a timed implementation, where tags correspond to real time. As we proceed through the intermediate levels, we can capture synchronization policies as additional constraints on the tags of different signals. This flexibility has a price: as in many cases, a very general model is denotational and not operational, i.e., the model offers little help in synthesis and other automatic transformations from one level of abstraction to another. In addition, because of its elegant simplicity, we must store all the intermediate models to keep track of the various transformations of the design.

In this paper we restrict ourselves to a less general model of tags that is suited to embedded and reactive systems, where each individual variable takes a totally ordered sequence of values.

In the original LSV model, parallel composition is classically based on the common notion of parallel composition by intersection. This means that two signals can be unified if they are identical (identical tags and identical data at each event); and two behaviours can be unified if they are identical. Then, unification is by superimposition. In this paper, we shall relax the request that unifiability corresponds to the equality of tags and we replace it with a notion of parameterizable unifiability that is modeled through a partial order relation. This becomes the key for capturing diverse models of computation.

2.2 Tagged Systems and Their (Homogeneous) Parallel Composition

If $(X, \leq_X)$ and $(Y, \leq_Y)$ are partial orders, $f \in X \mapsto Y$ is called *increasing* if $f(\leq_X) \subseteq \leq_Y$, i.e., $\forall x, x' \in X : x \leq_X x' \Rightarrow f(x) \leq_Y f(x')$.

Tag structures. A *tag structure* is a triple $(\mathcal{T}, \leq, \sqsubseteq)$, where $\mathcal{T}$ is a set of *tags,* and $\leq$ and $\sqsubseteq$ are two partial orders.

Partial order $\leq$ relates tags seen as *time stamps.* Call a *clock* any increasing function $(\mathbf{N}, \leq) \mapsto (\mathcal{T}, \leq)$.

Partial order $\sqsubseteq$, called the *unification order*, defines how to unify tags and is essential to express coordination among events. Write $\tau_1 \bowtie \tau_2$ to mean that there exists $\tau \in \mathcal{T}$ such that $\tau_i \sqsubseteq \tau$. We assume that any pair (τ_1, τ_2) of tags, such that $\tau_1 \bowtie \tau_2$ holds, possesses an upper bound. We denote it by $\tau_1 \sqcup \tau_2$. In other words, $(\mathcal{T}, \sqsubseteq)$ is a sup-semi-lattice. We call $\bowtie$ and $\sqcup$ the *unification relation* and *unification map,* respectively.

We assume that unification is causal with respect to partial order of time stamps: the result of the unification cannot occur prior in time than its constituents. Formally, if $\tau_1 \bowtie \tau_2$ is a unifiable pair then $\tau_i \leq (\tau_1 \sqcup \tau_2)$, for $i = 1, 2$. Equivalently:

$$\forall \tau, \tau' : \quad \tau \sqsubseteq \tau' \;\Rightarrow\; \tau \leq \tau'. \tag{1}$$

Condition (1) has the following consequence: if $\tau_1 \leq \tau_1'$, $\tau_2 \leq \tau_2'$, $\tau_1 \bowtie \tau_2$, and $\tau_1' \bowtie \tau_2'$ together hold, then $(\tau_1 \sqcup \tau_2) \leq (\tau_1' \sqcup \tau_2')$ must also hold. This ensures that the system obtained via parallel composition preserves the agreed order of its components.

Tagged systems. Let $\mathcal{V}$ be an underlying set of variables with domain D. For $V \subset \mathcal{V}$ finite, a V*-behaviour,* or simply behaviour, is an element:

$$\sigma \quad \in \quad V \mapsto \mathbf{N} \mapsto (\mathcal{T} \times D), \tag{2}$$

meaning that, for each $v \in V$, the n-th occurrence of v in behaviour σ has tag $\tau \in \mathcal{T}$ and value $x \in D$. For v a variable, the map $\sigma(v) \in \mathbf{N} \mapsto (\mathcal{T} \times D)$ is called a *signal.* For σ a behaviour, an *event* of σ is a tuple $(v, n, \tau, x) \in V \times \mathbf{N} \times \mathcal{T} \times D$ such that $\sigma(v)(n) = (\tau, x)$. Thus we can regard behaviours as sets of events. We require that, for each $v \in V$, the first projection of the map $\sigma(v)$ (it is a map $\mathbf{N} \mapsto \mathcal{T}$) is increasing with respect to $\leq$. Thus it is a clock and we call it the *clock of v in σ.* A *tagged system* is a triple $P = (V, \mathcal{T}, \Sigma)$, where V is a finite set of variables, $\mathcal{T}$ is a tag structure, and Σ a set of V-behaviours.

Homogeneous parallel composition. Consider two tagged systems $P_1 = (V_1, \mathcal{T}_1, \Sigma_1)$ and $P_2 = (V_2, \mathcal{T}_2, \Sigma_2)$ with identical tag structures $\mathcal{T}_1 = \mathcal{T}_2 = \mathcal{T}$. Let $\sqcup$ be the unification function of $\mathcal{T}$. For two events $e = (v, n, \tau, x)$ and $e' = (v', n', \tau', x')$, define

$$\begin{array}{lll} e \bowtie e' & \text{iff} & v = v', n = n', \tau \bowtie \tau', x = x', \text{ and} \\ e \bowtie e' & \Rightarrow & e \sqcup e' =_{\text{def}} (v, n, \tau \sqcup \tau', x). \end{array}$$

The unification map $\sqcup$ and relation $\bowtie$ extend pointwise to behaviours. Then, for σ a V-behaviour and σ' a V'-behaviour, define, by abuse of notation:

$$\sigma \bowtie \sigma' \text{ iff } \sigma_{|V \cap V'} \bowtie \sigma'_{|V \cap V'},$$

and then

$$\sigma \sqcup \sigma' \quad =_{\text{def}} \quad \left(\sigma_{|V \cap V'} \sqcup \sigma'_{|V \cap V'}\right) \cup \sigma_{|V \setminus V'} \cup \sigma'_{|V' \setminus V}.$$

where $\sigma_{|W}$ denotes the restriction of behaviour σ to the variables of W. Finally, for Σ and Σ' two sets of behaviours, define their *conjunction*

$$\Sigma \wedge \Sigma' \quad =_{\text{def}} \quad \left\{ \sigma \sqcup \sigma' \mid \sigma \in \Sigma, \sigma' \in \Sigma' \text{ and } \sigma \bowtie \sigma' \right\} \tag{3}$$

The *homogeneous parallel composition* of P_1 and P_2 is

$$P_1 \parallel P_2 \quad =_{\text{def}} \quad (V_1 \cup V_2, \mathcal{T}, \Sigma_1 \wedge \Sigma_2) \tag{4}$$

2.3 The Tagged System Model as an LSV Model?

At first glance, our model can be seen as a particular case of the LSV model. This is obtained by considering deterministic signals and composite tags with one special component called "natural tags":

Deterministic signals. In the original LSV setting a signal can take nondeterministically several values at a given tag. If we want to restrict to deterministic signals, the possible outcome at a given tag is either a single value or no value at all. We then move to $S = \mathcal{T} \mapsto D$, where, now, $\mapsto$ denotes partial maps.

Composite tags. When tags are composite, for instance $\mathcal{T} = \mathcal{T}_1 \times \mathcal{T}_2$, we get $S = \mathcal{T}_1 \times \mathcal{T}_2 \mapsto D$. This is known to be isomorphic to $S = \mathcal{T}_1 \mapsto (\mathcal{T}_2 \mapsto D)$.

Natural tags. Finally, taking $\mathcal{T}_1 = \mathbf{N}$ yields seemingly exactly our model of signals: $S = \mathbf{N} \mapsto (\mathcal{T} \mapsto D)$.

Discussion. Thus, our model seems less general than the LSV model in that we allow only for deterministic signals and tag sets of the special form $\mathcal{T} = \mathbf{N} \times \mathcal{T}_2$, where $\mathcal{T}_2$ is arbitrary.

However, this analysis is superficial and does not properly takes the underlying partial orders into account. The point is that the usual partial order associated with the Cartesian product of partial orders is the component-wise order:

$$(n_1, t_1) \leq (n_2, t_2) \Leftrightarrow (n_1 \leq n_2) \wedge (t_1 \leq t_2)$$

However, this composite order is symmetrical and this does not match the intent of our model: in our model, the order on $\mathbf{N}$ and the order on $\mathcal{T}_2$ do not play a symmetrical role: the former is local to each variable, whereas the latter is global to all variables. In particular, we have seen that taking $\mathcal{T}_2 = \mathbf{N}$ yields synchronous systems, where $\mathcal{T}_2$ indexes successive reactions.

Finally, taking a closer look reveals that our model cannot be considered simply as a special case of the LSV one.

2.4 Modeling with Tags

The concepts of tags allows us to express various models of computation as illustrated by the following examples. Further, tags facilitate the combination of diverse models of computation to model heterogeneous systems as discussed in Section 3.

We first discuss the case of parallel composition by intersection, with synchronous, asynchronous, and time-triggered systems as particular instances. Then, by making the unification rule for tags parameterized, we show how to deal with causality relations, scheduling constraints, and earliest execution times defined on each of the components.

Parallel composition by intersection of tagged systems corresponds to the following situation:

- the tag set $\mathcal{T}$ is arbitrary;
- the unification function is such that $\tau \bowtie \tau'$ iff $\tau = \tau'$, and $\tau \sqcup \tau' =_{\text{def}} \tau$.

Modeling synchronous systems, asynchronous systems, and timed systems, with this framework is extensively discussed in [5]. We summarize here the main points.

Synchrony. To represent *synchronous systems* with our model, take $\mathcal{T}_{\text{synch}} = \mathbf{N}$ as tag structure, and require that all clocks are strictly increasing. The tag index set $\mathcal{T}_{\text{synch}}$ organizes behaviours into successive reactions, as explained next. Call *reaction* a maximal set of events of σ with identical τ. Since clocks are strictly increasing, no two events of the same reaction can have the same variable. Regard a behaviour as a sequence of global reactions: $\sigma = \sigma_1, \sigma_2, \ldots$, with tags $\tau_1, \tau_2, \cdots \in \mathcal{T}_{\text{synch}}$. Thus $\mathcal{T}_{\text{synch}}$ provides a global, logical time basis.

Time-triggered systems. Timed systems such as those used to model Time-Triggered Architectures (TTA) [14] are similar to synchronous systems. The main difference is that the reaction index is replaced by physical real-time. Formally, the associated tag structure is $\mathcal{T}_{\text{tta}} =_{\text{def}} \mathbf{R}_+$ with its usual order $\leq$ for time stamping and the unification order is flat ($\tau \bowtie \tau'$ iff $\tau = \tau'$).

Asynchrony. The notion of asynchrony is vague. Any system that is not synchronous could be called asynchronous. However, we often want to restrict somewhat this notion to capture particular characteristics of the system in our modeling. In this paper, we take a very liberal interpretation for *asynchronous systems.* If we interpret a tag set as a constraint on the coordination of different signals of a system and the integer $n \in \mathbf{N}$ as the basic constraint of the sequence of events of the behaviour of a variable, then the most "coordination-unconstrained" system, i.e. the one with the highest degree of freedom in terms of choice of coordination mechanisms, could be considered an ideal asynchronous system. This translate into a model where the tag set does not give any information on the absolute or relative ordering of events. In a more formal way, to model asynchrony we choose $\mathcal{T} = \mathcal{T}_{\text{triv}} =_{\text{def}} \{\emptyset\}$, i.e. the trivial set consisting of a single element. Then, behaviours identify with elements $\sigma \in V \mapsto \mathbf{N} \mapsto D$. Hence, the behaviour of an asynchronous system is a set of sequences of values with each sequence associated to a distinct variable.

So far we discussed parallel composition by intersection. The following examples require a more sophisticated way to unify tags.

Causalities and scheduling specifications. Causalities or scheduling specifications were integral part of the original LSV model. Hence, it should be fairly simple to cast them in the tagged systems as well. We consider causality as a relation between tags of different signals that is in between the null relation among events of different signals in our notion of asynchronous systems and the existence of reactions that impose strong equality constraints among tags of different signals. The intent is, for example, to state that "the 2nd occurrence of x depends on the 3rd occurrence of b". Define $\mathbf{N}_0 =_{\text{def}} \mathbf{N} \cup \{0\}$. Define a *dependency* to be a map: $\delta = \mathcal{V} \mapsto \mathbf{N}_0$. We denote by $\mathcal{T}_{\text{dep}}$ the set of all dependencies, and we take $\mathcal{T}_{\text{dep}}$ as our tag structure. Thus an event has the form $e = (v, n, \delta, x)$, with the following interpretation: event e has v as associated variable, it is ranked n among the events with variable v, and it depends on the event of variable w that is ranked $\delta(w)$. The special case $\delta(w) = 0$ is interpreted as the absence of dependency. We take the convention that, for $e = (v, n, \delta, x)$ an event, $\delta(v) = n - 1$. Thus, on $\sigma(v)$, the set of dependencies reproduces the ranking. $\mathcal{T}_{\text{dep}}$ is equipped with the partial order defined by $\delta \leq \delta'$ iff $\forall v : \delta(v) \leq \delta'(v)$. Then we define the unification map $\sqcup$ for this case (note that $\sqsubseteq = \leq$):

$$\delta \sqcup \delta' \quad =_{\text{def}} \quad \max(\delta, \delta'). \tag{5}$$

With this definition, behaviours become labelled preorders as explained next. For σ a behaviour, and e, e' two events of σ, write:

$$e' \rightarrow_\sigma e \quad \text{iff} \quad \left\{ \begin{array}{rcl} e & = & (v, n, \delta, x) \\ e' & = & (v', n', \delta', x') \\ \delta(v') & = & n' \end{array} \right. \tag{6}$$

Note that when $n' > 0$ the condition $\delta(v') = n'$ makes this dependency effective. Definition (6) makes σ a labeled directed graph. Denote by $\preceq_\sigma$ the transitive and reflexive closure of $\rightarrow_\sigma$. It is a preorder [1].

Earliest execution times. To capture the earliest execution times of concurrent systems take $\mathcal{T}_{\text{date}} =_{\text{def}} \mathbf{R}_+$, the set of non-negative real numbers. Thus a tag $\tau \in \mathcal{T}_{\text{date}}$ assigns a date, and we define

$$\tau \sqcup \tau' \quad =_{\text{def}} \quad \max(\tau, \tau').$$

Hence, $\sqcup$ is here a total function. Composing two systems has the effect that the two components wait for the latest date of occurrence for each shared variable. For example, assume that variable v is an output of P and an input of Q in $P \parallel Q$. Then the earliest possible date of every event of variable v in Q is by convention 0, whereas each event associated to v has a certain date of production in P. In the parallel composition $P \parallel Q$, the dates of production by P prevail.

Compound tags. As discussed in the next sections, different tags can be combined by using the product of tag sets. This combination results into a compound, heterogeneous tag. For instance, one can consider synchronous systems that are timed and enhanced with causalities. Then, such systems can be "desynchronized", meaning that their reaction tag is erased, but their causality and time tags are kept.

3. HETEROGENEOUS SYSTEMS

In this section, we first define the algebra of tag structures. Then, we formally define heterogeneous systems based on this algebra. Heterogeneous systems are obtained via the parallel composition of heterogeneous components by means of communication media with appropriate tagged structures. We first define heterogeneous parallel composition for two components. Since this composition lacks associativity, the definition of composition for three or more subsystems is non trivial. To define general heterogeneous composition we resort to the use of morphisms in the tag algebra and a result on pullbacks from category theory.

3.1 The Algebra of Tag Structures

Tag structures will be denoted either explicitly by a triple $(\mathcal{T}, \leq, \sqsubseteq)$ = (set, partial order, unification order), or simply by the symbol $\mathcal{T}$ if the other items are understood.

Morphisms. Given two tag structures $\mathcal{T}, \mathcal{T}'$, call *morphism* a total map $\rho : \mathcal{T} \mapsto \mathcal{T}'$ which is increasing for

[1] We insist: "preorder", not "partial order"—this should not be a surprise, since the superposition of two partial orders generally yields a preorder.

$$\begin{tikzcd}
 & & \mathcal{T}_1 \arrow[dl, "\rho_{12}"'] \arrow[dr, "\rho'_{31}"] & & \\
 & \mathcal{S}_{12} & & \mathcal{S}_{31} & \\
\mathcal{T}_2 \arrow[ur, "\rho'_{12}"] \arrow[rr, "\rho_{23}"'] & & \mathcal{S}_{23} & & \mathcal{T}_3 \arrow[ul, "\rho_{31}"'] \arrow[ll, "\rho'_{23}"]
\end{tikzcd}$$

Figure 1: Morphism triangle.

$$\begin{tikzcd}
\mathcal{T}' \arrow[rr, "\rho'_2"] \arrow[dd, "\rho'_1"'] \arrow[dr] & & \mathcal{T}_2 \arrow[dd, "\rho_2"] \\
 & \mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2 \arrow[dr] & \\
\mathcal{T}_1 \arrow[rr, "\rho_1"'] & & \mathcal{T}
\end{tikzcd}$$

Figure 2: Pullback diagram for Lemma 1.

both orders $\leq$ and $\sqsubseteq$. Morphisms compose. Morphisms satisfy $\rho(\tau_1 \sqcup \tau_2) = \rho(\tau_1) \sqcup' \rho(\tau_2)$. Thus, morphisms preserve and possibly increase unifiability as they make more pairs of behaviours unifiable under parallel composition. As usual, $\mathcal{T}$ and $\mathcal{T}'$ are called *isomorphic* when there exist two morphisms $\rho : \mathcal{T} \mapsto \mathcal{T}'$ and $\rho' : \mathcal{T}' \mapsto \mathcal{T}$, such that $\rho' \circ \rho = Id_{\mathcal{T}}$ and $\rho \circ \rho' = Id_{\mathcal{T}'}$, where $\circ$ denotes the composition of functions. We do not distinguish isomorphic tag structures. Morphisms induce a preorder on tag structures:

$$\mathcal{T}' \preceq \mathcal{T} \quad \text{iff there exists a morphism} \quad \rho : \mathcal{T} \mapsto \mathcal{T}'.$$

Examples. Take $\mathcal{T} = \mathbf{R}_+, \mathcal{T}' = \mathbf{N}$ with $\bowtie$ being equality and $\leq$ being the usual order. The integer part $\mathbf{R}_+ \ni x \mapsto \lfloor x \rfloor \in \mathbf{N}$ is a morphism. More generally, sampling mechanisms are morphisms. Finally, a useful type of morphism is that of the canonical projections associated with products of tag structures, defined as follows: $(\mathcal{T}_1, \leq_1, \sqsubseteq_1) \times (\mathcal{T}_2, \leq_2, \sqsubseteq_2) =_{\rm def} (\mathcal{T}_1 \times \mathcal{T}_2, \leq_1 \times \leq_2, \sqsubseteq_1 \times \sqsubseteq_2)$.

Desynchronization. Given the tag structures $\mathcal{T} = \mathcal{T}_{\rm synch}$ and $\mathcal{T}' = \mathcal{T}_{\rm triv}$, the morphism $\rho : \mathcal{T} \mapsto \mathcal{T}'$ that maps each tag $\tau \in \mathcal{T}$ to the unique tag constituting $\mathcal{T}'$ is called the *desynchronization morphism.*

Desynchronization is of great interest in practical applications and, generally , it corresponds to the operation of making the synchronization constraints less stringent. Since synchronization constraints are captured by a tag structure that forces events to belong to reactions, desynchronization corresponds to remove, at least partially, the reaction tags. In the above definition this removal is complete and corresponds to mapping the tag structure of a synchronous system into the tag structure of an asynchronous system. In an interesting case this mapping corresponds to the task of deriving an asynchronous and very efficient (minimal overhead) implementation for a synchronous specification, which captures the behaviour of the system that we want to build. In general, it is rare that this radical desynchronization has the same behaviour of the specification. Hence, we are interested in the case where we can map the specification into a partially desynchronized system. Distributed architectures of practical interest have this characteristic.

Fibred product. For $\mathcal{T}_1 \xrightarrow{\rho_1} \mathcal{T} \xleftarrow{\rho_2} \mathcal{T}_2$ two morphisms, define the *fibred product*:

$$\mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2 =_{\rm def} \{ (\tau_1, \tau_2) \in \mathcal{T}_1 \times \mathcal{T}_2 \mid \rho_1(\tau_1) = \rho_2(\tau_2) \} \quad (7)$$

also written $\mathcal{T}_1 \times_{\mathcal{T}} \mathcal{T}_2$. The fibred product is equipped with the restriction of the product orders $\leq_1 \times \leq_2$ and $\sqsubseteq_1 \times \sqsubseteq_2$. The map $\rho : (\tau_1, \tau_2) \in \mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2 \mapsto \rho_1(\tau_1) \in \mathcal{T}$ defines the canonical morphism associated with this fibred product.

Examples. Note that $\mathcal{T} \;{}_{Id}\!\times_{Id}\; \mathcal{T} = \mathcal{T}$. A more interesting case is when $\mathcal{T}_i = \mathcal{T}'_i \times \mathcal{T}, i = 1, 2$, and the two morphisms are the projections $\pi_i : \mathcal{T}_i \mapsto \mathcal{T}$. Then, $\mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2$ is the product $\mathcal{T}'_1 \times \mathcal{T} \times \mathcal{T}'_2$.

Multiple fibred product. Formula (7) defines the fibred product for two tag structures. How can we generalize it for more than two? For instance, consider the "triangle" shown in Fig. 1. This induces several fibred products. For instance:

$$(\mathcal{T}_1 \times_{\mathcal{S}_{12}} \mathcal{T}_2) \times_{\mathcal{S}_{23}} (\mathcal{T}_3 \times_{\mathcal{S}_{31}} \mathcal{T}_1)$$
and
$$((\mathcal{T}_1 \times_{\mathcal{S}_{12}} \mathcal{T}_2) \times_{\mathcal{S}_{23}} \mathcal{T}_3) \times_{\mathcal{S}_{31}} \mathcal{T}_1.$$

Lemma 1 below related to ([11]–ch. 3, "pullback lemma"), ensures that

these seemingly different fibred products
are in fact all isomorphic. (8)

The same holds for every network of tags sets like in Fig. 1. This fact is essential to the modeling of heterogeneous architectures that we discuss in the next section.

The following lemma says that "the fibred product (7) defines a *pullback* [11] in the category of tag structures". This result is essential in defining the heterogeneous composition of more than two components.

Lemma 1. *For any two morphisms $\rho_i : \mathcal{T}_i \mapsto \mathcal{T}, i = 1, 2$, let $(\mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2)$ be their fibred product and ρ the associated canonical morphism. Then, for every two morphisms $\rho'_i : \mathcal{T}' \mapsto \mathcal{T}_i, i = 1, 2$ such that $\rho_1 \circ \rho'_1 = \rho_2 \circ \rho'_2$, there exists a unique morphism $\rho' : \mathcal{T}' \mapsto (\mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2)$ such that $\rho_i \circ \rho'_i = \rho \circ \rho'$, for $i = 1, 2$.*

Proof. First observe that pullbacks are defined in the category of sets exactly by using formula (7). But objects and morphisms in the category of tag structures induce objects and morphisms in the category of sets, by ignoring the partial order structure and unification orders on tag sets and related morphisms. This shows the existence of ρ' as a function satisfying the above factorizations. It remains to see that ρ' is in fact a morphism of tag structures, i.e., that ρ': 1/ is increasing, and 2/ commutes with the unification order. This is immediate.

3.2 Heterogeneous Parallel Composition

In this section we define the composition of two tagged systems $P_1 = (V_1, \mathcal{T}_1, \Sigma_1)$ and $P_2 = (V_2, \mathcal{T}_2, \Sigma_2)$ when $\mathcal{T}_1 \neq \mathcal{T}_2$.

Given a morphism $\rho : \mathcal{T} \mapsto \mathcal{T}'$ and a behaviour $\sigma \in V \mapsto \mathbf{N} \mapsto (\mathcal{T} \times D)$, replacing τ by $\rho(\tau)$ in σ defines a new behaviour having $\mathcal{T}'$ as tag structure. This behaviour is denoted as σ_ρ, or (with some abuse of notation) as $\sigma \circ \rho$.

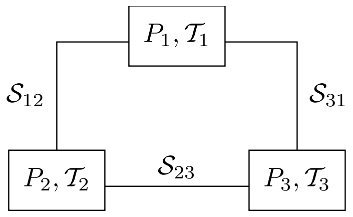

Figure 3: *A graphical notation for heterogeneous systems.*

Performing this for every behaviour of a tagged system P with tag structure $\mathcal{T}$ yields the tagged system

$$P_\rho, \text{ also denoted by } P_{\mathcal{T}'}. \tag{9}$$

Assume two morphisms $\mathcal{T}_1 \xrightarrow{\rho_1} \mathcal{T} \xleftarrow{\rho_2} \mathcal{T}_2$. Write: $\sigma_1 \;{}_{\rho_1}\!\bowtie_{\rho_2}\; \sigma_2$ iff $\sigma_1 \circ \rho_1 \bowtie \sigma_2 \circ \rho_2$. For (σ_1, σ_2) a pair satisfying $\sigma_1 \;{}_{\rho_1}\!\bowtie_{\rho_2}\; \sigma_2$, define

$$\sigma_1 \;{}_{\rho_1}\!\sqcup_{\rho_2}\; \sigma_2 \tag{10}$$

as being the set of events $(v, n, (\tau_1, \tau_2), x)$ such that $\rho_1(\tau_1) = \rho_2(\tau_2) =_{\text{def}} \tau$ and the followings hold:

1. if $v \in V_i$, $i \in \{1, 2\}$, then (v, n, τ_i, x) is an event of σ_i.
2. (v, n, τ, x) is an event of $\sigma_1 \circ \rho_1 \sqcup \sigma_2 \circ \rho_2$.

We are now ready to define the *heterogeneous conjunction* of Σ_1 and Σ_2 by:

$$\Sigma_1 \;{}_{\rho_1}\!\wedge_{\rho_2}\; \Sigma_2 \quad =_{\text{def}} \quad \{\sigma_1 \;{}_{\rho_1}\!\sqcup_{\rho_2}\; \sigma_2 \mid \sigma_1 \;{}_{\rho_1}\!\bowtie_{\rho_2}\; \sigma_2\} \tag{11}$$

Finally, the *heterogeneous parallel composition* of P_1 and P_2 is defined as

$$P_1 \;{}_{(\rho_1}\!\|_{\rho_2)}\; P_2 = (\, V_1 \cup V_2 \,,\, \mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2 \,,\, \Sigma_1 \;{}_{\rho_1}\!\wedge_{\rho_2}\; \Sigma_2 \,) \tag{12}$$

For convenience, when morphisms ρ_1 and ρ_2 are understood, we prefer the following notation instead of (12), where $\mathcal{T}$ is such that $\mathcal{T}_1 \xrightarrow{\rho_1} \mathcal{T} \xleftarrow{\rho_2} \mathcal{T}_2$:

$$P_1 \;\|_{\mathcal{T}}\; P_2 \tag{13}$$

Examples: GALS and hybrid timed/yntimed systems. (The reader is referred to Section 2.4 for the below mentioned tag structures.)

For GALS, take $\mathcal{T}_1 = \mathcal{T}_2 = \mathcal{T}_{\text{synch}}$, where $\mathcal{T}_{\text{synch}} = \mathbf{N}$ is the tag structure of synchronous systems, and consider $P_1 \;\|_{\mathcal{T}_{\text{triv}}}\; P_2$, where $\mathcal{T} = \mathcal{T}_{\text{triv}}$ is the tag structure of asynchronous ones.

To model a synchronous system $P = (V, \mathcal{T}_{\text{synch}}, \Sigma)$ interacting with its asynchronous environment $A = (W, \mathcal{T}_{\text{triv}}, \Sigma')$, take the heterogeneous composition $P \;\|_{\mathcal{T}_{\text{triv}}}\; A$.

To model the composition of timed systems with untimed systems, consider the heterogeneous system $P \;\|_{\mathcal{T}_{\text{synch}}}\; Q$, where $P = (V, \mathcal{T}_{\text{synch}} \times \mathcal{T}_{\text{date}}, \Sigma)$ is a synchronous timed system while $Q = (W, \mathcal{T}_{\text{synch}}, \Sigma')$ is a synchronous but untimed system.

3.3 Heterogeneous Systems and Architectures

So far we only defined heterogeneous parallel composition of two components. To capture more complex architectures we need to define it for a heterogeneous network of components. Fact (8) will be instrumental in doing this. The expression

$$P_1 \;\|_{\mathcal{S}_{12}}\; P_2 \;\|_{\mathcal{S}_{23}}\; P_3 \;\|_{\mathcal{S}_{31}}\; P_1, \tag{14}$$

where the repeated symbol P_1 refers to the same component, gives raise to several possible interpretations. We give only two of them:

$$\begin{array}{c} (P_1 \;\|_{\mathcal{S}_{12}}\; P_2) \;\|_{\mathcal{S}_{23}}\; (P_3 \;\|_{\mathcal{S}_{31}}\; P_1) \\ \text{and} \\ ((P_1 \;\|_{\mathcal{S}_{12}}\; P_2) \;\|_{\mathcal{S}_{23}}\; P_3) \;\|_{\mathcal{S}_{31}}\; P_1. \end{array} \tag{15}$$

Thanks to Fact (8), the two interpretations yield isomorphic tagged systems, i.e., tagged systems that are equal up to an isomorphism between their tag structures. This property is a restricted form of associativity [2]. It ensures that *expression (14) is well defined.*

Fig. 3 proposes a graphical notation for heterogeneous systems. In this figure, pair $(P_1, \mathcal{T}_1)$ specifies that the corresponding box refers to system P_1 having tag structure $\mathcal{T}_1$. The label $\mathcal{S}_{12}$ sitting aside the link between $(P_1, \mathcal{T}_1)$ and $(P_2, \mathcal{T}_2)$ denotes a tag structure such that $\mathcal{S}_{12} \preceq \mathcal{T}_i, i = 1, 2$. The presence of this label indicates that these two systems are composed via $P_1 \;\|_{\mathcal{S}_{12}}\; P_2$. In general, the components of the considered system are indexed by some set I (in Fig. 3, $I = \{1, 2, 3\}$). Collect the tag structures of the different communication media into the matrix $\mathbf{S} =_{\text{def}} (\mathcal{S}_{ij})_{(i,j) \in I \times I}$. The heterogeneous architecture shown in Fig. 3 is denoted by

$$\mathbf{P} = \|_{\mathbf{S}, i \in I}\, P_i, \quad \text{or simply} \quad \mathbf{P} = \|_{i \in I}\, P_i, \tag{16}$$

if $\mathbf{S}$ is understood. For $\mathbf{P}$ as in (16), a tag structure $\mathcal{T}$ is called $\mathbf{P}$*-consistent* iff $\mathcal{T} \preceq \mathcal{S}_{ij}$ for every pair (i, j). Given a $\mathbf{P}$-consistent tag structure $\mathcal{T}$, denote by

$$\mathbf{P}_{\mathcal{T}} \tag{17}$$

the homogeneous tagged system having the same set of variables as $\mathbf{P}$, tag structure $\mathcal{T}$, and a set of behaviours obtained by mapping all tags to $\mathcal{T}$. This is well defined since, for every j, $\mathcal{T}_i \succeq \mathcal{S}_{ij} \succeq \mathcal{T}$.

4. CORRECT DEPLOYMENT

In this section we apply our framework to the formalization of the practically important—but generally informal—requirement of "correct deployment".

4.1 Preserving Semantics: Formalization

The situation is illustrated in Fig. 4. Diagram **(a)** depicts a homogeneous specification $P_1 \| P_2$, where $P_1 = (V_1, \mathcal{T}, \Sigma_1)$ and $P_2 = (V_2, \mathcal{T}, \Sigma_2)$ possess identical tag structure $\mathcal{T}$. Let $W =_{\text{def}} V_1 \cap V_2$ be the set of shared variables of P_1 and P_2. To prepare for deployment, we wish to distinguish the shared variables when seen from P_1 or P_2. Formally, let W_1 and W_2 be two distinct copies of W, rename each shared variable w of P_1 by $w_1 \in W_1$, and similarly for P_2. This renaming is modeled by the "identity" channel with tag structure $\mathcal{T}$, implementing $w_1 = w_2$ for each $w \in W$: $Id = (W_1 \uplus W_2, \mathcal{T}, \Sigma)$, where Σ is the set of behaviours such that, for each $w \in W$, the signals associated with w_1 and w_2 are equal—the homogeneous system $P_1 \| Id \| P_2$ is depicted in Diagram **(b)**.

[2] Full fledged associativity cannot be considered, however: the first expression of (15) implicitly assumes for $\mathcal{S}_{31}$ that $\mathcal{S}_{31} \preceq \mathcal{T}_3$ and $\mathcal{S}_{31} \preceq \mathcal{T}_1$; on the other hand, the second expression of (15) requires the strictly weaker conditions that $\mathcal{S}_{31} \preceq ((\mathcal{T}_1 \times_{\mathcal{S}_{12}} \mathcal{T}_2) \times_{\mathcal{S}_{23}} \mathcal{T}_3)$ and $\mathcal{S}_{31} \preceq \mathcal{T}_1$.

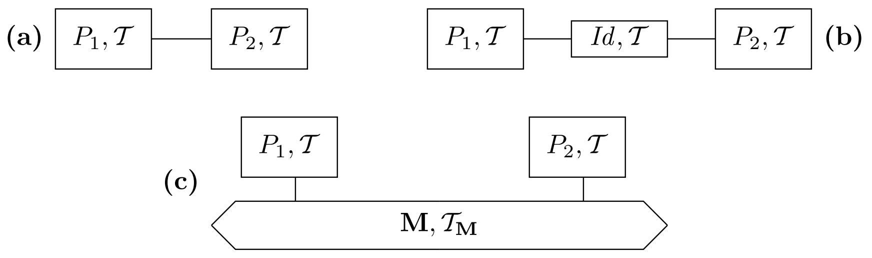

Figure 4: ***A specification and its actual deployment:*** **(a)** specification, **(b)** same but with an explicit "identity" channel, **(c)** the deployment over a (possibly heterogeneous) communication medium with tag family $\mathbf{T}$.

When deploying the specification, the identity channel is replaced by a communication medium $\mathbf{M}$, which is a (possibly heterogeneous) tagged system as in (16). Two semantics can be considered:

$$\begin{array}{rccl} \text{the specification semantics} & : & \mathbf{S} & = & P_1 \parallel Id \parallel P_2 \\ \text{the deployment semantics} & : & \mathbf{D} & = & P_1 \parallel \mathbf{M} \parallel P_2 \end{array} \tag{18}$$

where the latter involve morphisms since $P_i, i = 1, 2$ on the one hand, and $\mathbf{M}$ on the other hand, possess in general different tag structures. In addition, $\mathbf{M}$ may be a heterogeneous system like in (16).

Definition 1. *$P_1 \parallel \mathbf{M} \parallel P_2$ simulates $P_1 \parallel Id \parallel P_2$, written*

$$P_1 \parallel \mathbf{M} \parallel P_2 \overset{\leq}{\equiv} P_1 \parallel Id \parallel P_2,$$

iff each pair of behaviours unifiable in the deployment is also unifiable in the specification, i.e. when $\forall(\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2$, (i) $\Rightarrow$ (ii) holds, where:

(i) $\exists \sigma' \in \Sigma_{\mathbf{D}}$ *s.t.* $\mathbf{proj}_{V_1}(\sigma') = \sigma_1$ *and* $\mathbf{proj}_{V_2}(\sigma') = \sigma_2$
(ii) $\exists \sigma \in \Sigma_{\mathbf{S}}$ *s.t.* $\mathbf{proj}_{V_1}(\sigma) = \sigma_1$ *and* $\mathbf{proj}_{V_2}(\sigma) = \sigma_2$

$P_1 \parallel \mathbf{M} \parallel P_2$ is semantics preserving *with respect to $P_1 \parallel Id \parallel P_2$, written*

$$P_1 \parallel \mathbf{M} \parallel P_2 \quad \equiv \quad P_1 \parallel Id \parallel P_2 \tag{19}$$

iff $P_1 \parallel \mathbf{M} \parallel P_2 \overset{\leq}{\equiv} P_1 \parallel Id \parallel P_2$ and $P_1 \parallel \mathbf{M} \parallel P_2 \overset{\geq}{\equiv} P_1 \parallel Id \parallel P_2$ hold.

For $\mathcal{T}' \prec \mathcal{T}$ and $\rho : \mathcal{T} \mapsto \mathcal{T}'$, the heterogeneous parallel composition $P_1 \parallel_{\mathcal{T}'} P_2$ can be seen as a particular case of deployment: we simply write

$$P_1 \parallel_{\mathcal{T}'} P_2 \quad \equiv \quad P_1 \parallel P_2 \tag{20}$$

iff $P_1 \parallel (Id, \mathcal{T}') \parallel P_2 \equiv P_1 \parallel Id \parallel P_2$ holds (note the heterogeneous parallel composition on the left hand side).

4.2 General Results on Correct Deployment

Analyzing requirement (20) Here we investigate conditions ensuring (20). See (9) for the notation $P_{\mathcal{T}'}$ used in the following theorem:

Theorem 1. *The pair (P_1, P_2) satisfies condition (20) if it satisfies the following two conditions:*

$$\forall i \in \{1,2\} : (P_i)_{\mathcal{T}'} \text{ is in bijection with } P_i \tag{21}$$
$$(P_1 \parallel P_2)_{\mathcal{T}'} = (P_1)_{\mathcal{T}'} \parallel (P_2)_{\mathcal{T}'} \tag{22}$$

Comment. The primary application of this general theorem is when P and Q are synchronous systems, and $\mathcal{T}' = \mathcal{T}_{\text{triv}}$ is the tag structure for asynchrony. This formalizes GALS deployment. Thus, Theorem 1 provides sufficient conditions to ensure correct GALS deployment.

Proof. It is a mild variation of the proof of Theorem 1 of [5]. The detailed proof with the present framework will be found in the extended version [4] of this paper. $\diamond$

Corollary 1. *Let P_1 and P_2 be synchronous systems whose behaviours are equipped with some equivalence relation $\sim$, and assume that P_1 and P_2 are closed with respect to $\sim$. Then, the pair (P_1, P_2) satisfies condition (19) if it satisfies the following two conditions:*

$$\forall i \in \{1,2\} : (P_i)_{\mathcal{T}'} \text{ is in bijection with } P_i/\sim \tag{23}$$
$$(P_1 \parallel P_2)_{\mathcal{T}'} = (P_1)_{\mathcal{T}'} \parallel (P_2)_{\mathcal{T}'} \tag{24}$$

where $P_i/\sim$ is the quotient of P_i modulo $\sim$.

Comment. This result is of particular interest when $\sim$ is the equivalence modulo stuttering, defined in [5].

Proof. A mild variation of the proof of Corollary 1 of [5]. A detailed proof will be found in [4].

Analyzing requirement (19). Here we investigate conditions ensuring (19). Using notation (16), decompose the communication medium as $\mathbf{M} = \parallel_{\mathbf{T}, i \in I} M_i$. Let $\mathcal{T}_{\mathbf{M}}$ be a $(P_1 \parallel \mathbf{M} \parallel P_2)$-consistent tag structure, and let $\mathcal{T}$ be the common tag structure of P_i, for $i = 1, 2$. Note that $\mathcal{T} \succeq \mathcal{T}_{\mathbf{M}}$. The following theorem complements Theorem 1 and its corollary:

Theorem 2. *Triple $(P_1, \mathbf{M}, P_2)$ satisfies condition (19) if it satisfies the following two conditions:*

$$P_1 \parallel_{\mathcal{T}_{\mathbf{M}}} P_2 \equiv P_1 \parallel P_2 \tag{25}$$
$$\mathbf{M} \textit{ is in bijection with } \mathbf{M}_{\mathcal{T}_{\mathbf{M}}}, \textit{ and } \mathbf{M}_{\mathcal{T}_{\mathbf{M}}} = (Id, \mathcal{T}_{\mathbf{M}}) \tag{26}$$

where $\mathbf{M}_{\mathcal{T}_{\mathbf{M}}}$ is defined in (17) and equality in (26) means that the two systems possess identical sets of behaviours when restricted to the variables of P_1 or P_2 and local variables of $\mathbf{M}$ being hidden.

Comment. Theorem 2 is a *"separation theorem"*: condition (25) does (almost) not involve the communication medium, since only the greatest lower tag structure $\mathcal{T}_{\mathbf{M}}$ and associated morphism $\rho : \mathcal{T} \mapsto \mathcal{T}_{\mathbf{M}}$ play a role. On the other hand, condition (26) involves only the medium, not the application. Note that condition (25) is handled by Theorem 1 or Corollary 1.

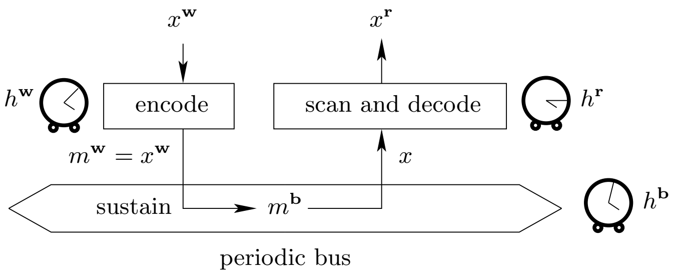

Figure 5: *The* LTTA*-protocol.*

PROOF. First, note that, by using the renaming convention of (18), condition (25) rewrites $P_1 \parallel_{\mathcal{T}_{\mathbf{M}}} (Id, \mathcal{T}_{\mathbf{M}}) \parallel_{\mathcal{T}_{\mathbf{M}}} P_2 \equiv P_1 \parallel Id \parallel P_2$. Thus

$$P_1 \parallel_{\mathcal{T}_{\mathbf{M}}} \mathbf{M}_{\mathcal{T}_{\mathbf{M}}} \parallel_{\mathcal{T}_{\mathbf{M}}} P_2 \quad \equiv \quad P_1 \parallel Id \parallel P_2 \tag{27}$$

follows, by the second statement of (26). Next, it is always true that $P_1 \parallel \mathbf{M} \parallel P_2$ simulates $P_1 \parallel_{\mathcal{T}_{\mathbf{M}}} \mathbf{M}_{\mathcal{T}_{\mathbf{M}}} \parallel_{\mathcal{T}_{\mathbf{M}}} P_2$, i.e.,

$$P_1 \parallel \mathbf{M} \parallel P_2 \quad \overset{\leq}{\equiv} \quad P_1 \parallel_{\mathcal{T}_{\mathbf{M}}} \mathbf{M}_{\mathcal{T}_{\mathbf{M}}} \parallel_{\mathcal{T}_{\mathbf{M}}} P_2, \tag{28}$$

Recall Definition 1 of semantics preserving. Then, (27,28) and the first statement of (26) together complete the proof.

5. DEPLOYING TIMED SYNCHRONOUS SPECIFICATIONS OVER LTTA

Loosely Time-Triggered Architectures (LTTA) were introduced in [6] as a less constrained version of H. Kopetz' TTA [14]. The reader is referred to references [6, 7] for the motivations behind LTTA. In this section, we provide the complete foundations for correct-by-construction deployment over LTTA. This complements the partial analysis provided in [6]. The reader may find that the analysis provided to support LTTA looks straightforward. However one should remember that, when this research goal was proposed in [8], it was considered very hard to formally understand the engineering practice. Hence, an intrinsic value of our approach consists also of casting the problem into a framework where the fundamental issues emerge with clarity.

5.1 The LTTA Architecture

The LTTA protocol is illustrated in Figure 5 where the three watches indicate a different time (*they are not synchronized*). We consider three devices, the *writer,* the *bus,* and the *reader,* indicated by the superscripts $(.)^{\mathbf{w}}$, $(.)^{\mathbf{b}}$, and $(.)^{\mathbf{r}}$, respectively. Each device is activated by its own, approximately periodic, clock. The different clocks are *not* synchronized. In the following specification, the different sequences written, fetched, or read, are indexed by the set $\mathbf{N} = \{1, 2, 3, \ldots, n, \ldots\}$ of natural integers, and we reserve the index 0 for the initial conditions, whenever needed. Set $\mathbf{N}$ will serve to index the successive events of each individual signal, exactly as in our model of Section 2.2.

The writer: At the time $t^{\mathbf{w}}(n)$ of the n-th tick of his clock, the writer generates a new value $x^{\mathbf{w}}(n)$ it wants to communicate and stores it in its private output buffer. Thus at any time t, the writer's output buffer content $m^{\mathbf{w}}$ is the last value that was written into it, that is the one with the largest index whose tick occurred before t:

$$m^{\mathbf{w}}(t) = x^{\mathbf{w}}(n)\,,\ \text{where}\ n = \sup\{n' \mid t^{\mathbf{w}}(n') < t\} \tag{29}$$

The bus: At the time $t^{\mathbf{b}}(n)$ of its n-th clock tick, it fetches the value in the writer's output buffer and stores it, immediately after, in the reader's input buffer. Thus, at any time t, the reader's input buffer content offered by the bus, denote it by $m^{\mathbf{b}}$, is the last value that was written into it, i.e., the one written at the latest bus clock tick preceding t:

$$m^{\mathbf{b}}(t) = m^{\mathbf{w}}(t^{\mathbf{b}}(n))\,,\ \text{where}\ n = \sup\{n' \mid t^{\mathbf{b}}(n') < t\} \tag{30}$$

The reader: At the time $t^{\mathbf{r}}(n)$ of its n-th clock tick, it copies the value of its input buffer into its output variable $x^{\mathbf{r}}(n)$:

$$x^{\mathbf{r}}(n) = m^{\mathbf{b}}(t^{\mathbf{r}}(n)) \tag{31}$$

The protocol defined by formulas (29,30,31) is called LTTA-*protocol.*

5.2 A Formal Tagged System Model of LTTA

In this section, we study the preservation of semantics for multiple-channels, multiple-clocked synchronous systems.

The problem. The situation is illustrated on Fig. 6. In this figure, we show two multiple-clocked synchronous systems P and Q. The original model of communication is that of synchronous broadcast, in which tag is reaction index; this is shown in **(a)** and in **(b)** by making the identity channel explicit. The actual deployment is by means of the LTTA bus with its associated buffers, in which tag is physical time, this is shown in **(c)**. In diagram **(c)** we show also the different tag structures involved in the LTTA medium. In the sequel, we shall denote by $\mathcal{L}$ the tagged system model of LTTA.

Since the trivial tag structure $\mathcal{T}_{\text{triv}}$ associated to full asynchrony satisfies $\mathcal{T}_{\text{triv}} \preceq \mathcal{T}_{\text{synch}}$ and $\mathcal{T}_{\text{triv}} \preceq \mathcal{T}_{\text{tta}}$ and no other "known" tag structure satisfies these two conditions, $\mathcal{T}_{\text{triv}}$ is a natural candidate for a $\mathcal{L}$-consistent tag structure. Informally, switching from logical time to physical time, and then back to logical time, destroys synchronization information. It is therefore non trivial that LTTA deployment can preserve semantics.

The tagged system $\mathcal{L}^{w\to r}$. More precisely, our specification is $P \parallel Q$, with $\parallel$ the synchronous parallel composition. The set of shared variables is $V_P \cap V_Q$. For convenience we shall distinguish the instance of $v \in V_P \cap V_Q$ that is attached to P by calling it v_P, and similarly for v_Q. Since the physical communication through LTTA medium takes time, it makes sense assuming that, for each individual shared variable, communication is directed, i.e., for every pair (v_P, v_Q), one of the two is an output and the other is an input. Suppose that v_P is an output of P, then the parallel composition can be re-interpreted as the directed assignment $v_Q := v_P$. This assignment can thus be seen as an instance of the generic single-channel communication $r := w$, where symbols r and w refer to *reader* and *writer*, respectively. We denote this ideal single-channel communication model by $Id^{w\to r}$. Its actual deployment over LTTA is denoted by $\mathcal{L}^{w\to r}$. The latter decomposes as:

$$\mathcal{L}^{w\to r} \quad = \quad P^{\mathbf{w}} \parallel P^{\mathbf{b}} \parallel P^{\mathbf{r}}. \tag{32}$$

As the reader will see, (32) is a heterogeneous parallel composition, since the components of this parallel composition possess different tag structures. These components are:

$P^{\mathbf{w}}$ is the writer buffer: The writer buffer is a hybrid synchronous/timed tagged system, described as follows.

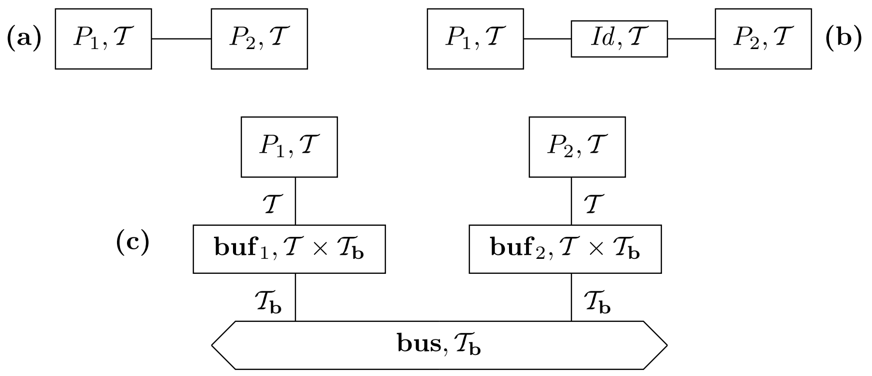

Figure 6: Ltta ***deployment depicted as in Fig. 4, with the conventions of Fig. 3.*** In all diagrams, $\mathcal{T} = \mathcal{T}_{\mathrm{synch}} = \mathbf{N}$ captures logical time and $\mathcal{T}_\mathbf{b} = \mathcal{T}_{\mathrm{tta}} = \mathbf{R}_+$ models physical time from time-triggered systems, as introduced in Section 2.4.

- *tag structure:* $\mathcal{T}^\mathbf{w} = \mathcal{T}_{\mathrm{synch}} \times \mathcal{T}_{\mathrm{tta}}$ = *logical time* $\times$ *physical time of* TTA. The former carries the synchronous semantics with its successive reactions, whereas the latter carries the timed semantics.

- *Variables:* the writer has a single variable w. The logical clock (in the synchronous sense) of w is a subclock of the activation clock of $P^\mathbf{w}$.

- *Behaviours:*

$$\begin{array}{rcl} \sigma(w)(k) & = & ((m_k, t_{m_k}), x_k), \\ & \text{where} & t_m = \lambda^\mathbf{w} m + \varphi^\mathbf{w} \end{array}$$

where $(\lambda^\mathbf{w}, \varphi^\mathbf{w})$ = (period, phase) of the writer buffer periodic clock. In (33), m_k is the index of the reaction $\sigma(w)(k)$ belongs to, where $m = 1, 2, \ldots$ counts the reactions of the buffer—the map $k \mapsto m_k$ is strictly increasing. Then, t_m is the physical date of the m-th reaction of the buffer.

$P^\mathbf{r}$ is the reader buffer: Same comments as for the writer buffer.

- *tag structure:* $\mathcal{T}^\mathbf{r} = \mathcal{T}_{\mathrm{synch}} \times \mathcal{T}_{\mathrm{tta}}$ = *logical time* $\times$ *physical time of* TTA.

- *Variables:* the reader has a single variable r

- *Behaviours:*

$$\sigma(r)(\ell) \;=\; ((p_\ell, t_{p_\ell}), x_\ell), \text{ where } t_p = \lambda^\mathbf{r} p + \varphi^\mathbf{r}$$

where $(\lambda^\mathbf{r}, \varphi^\mathbf{r})$ = (period, phase) of the reader buffer periodic clock.

$P^\mathbf{b}$ is the bus: The bus is a purely timed system, described as follows. For $e = (\tau, x)$ a pair (tag, value), we denote by $x[e]$ the value carried by e.

- *tag structure:* $\mathcal{T}^\mathbf{b} = \mathcal{T}_{\mathrm{tta}}$ = *physical time of* TTA.

- *Variables:* the bus has three variables w, ξ, r, where ξ is local.

- *Behaviours:*

$$\begin{array}{rcl} \sigma(w)(k) & = & (t^w_k, x_k) \\ \sigma(\xi)(n) & = & (t^\xi_n, x[\sigma(w)(k_n)]), \\ & \text{where} & k_n = \max\{k \,|\, t^w_k < t^\xi_n\} \\ \sigma(r)(n) & = & (t^r_\ell, x[\sigma(\xi)(n_\ell)]), \\ & \text{where} & n_\ell = \max\{n \,|\, t^\xi_n < t^r_\ell\} \end{array}$$

and $t^\xi_n = \lambda^\mathbf{b} n + \varphi^\mathbf{b}$, $(\lambda^\mathbf{b}, \varphi^\mathbf{b})$ = (period, phase) of the bus periodic clock.

5.3 Conditions for Correct-by-Construction Deployment over Ltta

Formal modeling of deployment. We consider the two synchronous systems P and Q. For $v \in V_P \cap V_Q$ a shared variable, we write v_P (resp. v_Q) when referring to its instance in P (resp. Q). Then, out_P denotes the set of outputs of P.

The specification semantics S. It is given by

$$P \parallel \underbrace{\left(\|_{v\in \mathrm{out}_P \cap V_Q}\; Id^{\,v_P\to v_Q}\right) \parallel \left(\|_{v\in V_P \cap \mathrm{out}_Q}\; Id^{\,v_Q\to v_P}\right)}_{Id\text{: a bundle of directed synchronous identity channels}} \parallel Q$$

Here $\mathcal{S}$ is a (purely) synchronous system.

The deployment semantics D. It is given by

$$P \parallel \underbrace{\left(\|_{v\in \mathrm{out}_P \cap V_Q}\; \mathcal{L}^{\,v_P\to v_Q}\right) \parallel \left(\|_{v\in V_P \cap \mathrm{out}_Q}\; \mathcal{L}^{\,v_Q\to v_P}\right)}_{\mathcal{L}\text{: a bundle of directed Ltta channels}} \parallel Q$$

Here, **D** is a hybrid system consisting of two synchronous systems P and Q communicating via synchronous/timed system $\mathcal{L}$. This model is somewhat cheated. It implicitly assumes that a bundle of Ltta channels is indeed available, meaning that a new bus should be assigned to each peer communication. In practice, only one bus is available and the different peer communications are multiplexed. But this time-division multiplexing is easily handled by a proper choice of the pair (period, phase).

Preserving semantics. We shall use the general results of Section 4. Our communication medium is $\mathcal{L}$, it is a heterogeneous tagged system that is a mix of timed/synchronous and purely timed system. On the other hand, the application for deployment is purely synchronous. Therefore, the trivial tag set $\mathcal{T}_{\mathrm{triv}}$ is the natural candidate for a **D**-consistent tag structure. Using Theorem 2, we derive the following sufficient conditions for the Ltta deployment to preserve semantics:

$$P_1 \parallel_{\mathcal{T}_{\rm triv}} P_2 \equiv P_1 \parallel P_2 \quad (33)$$

$$\mathcal{L} \text{ is in bijection with } \mathcal{L}_{\mathcal{T}_{\rm triv}}, \text{ and } \mathcal{L}_{\mathcal{T}_{\rm triv}} = (Id, \mathcal{T}_{\rm triv}) \quad (34)$$

Condition (33) involves only the robustness of deploying the pair (P_1, P_2) over a GALS architecture, it does not depend on Ltta. Condition (34) involves only Ltta, not the considered application. Since condition (33) has already been addressed elsewhere [2, 3, 16], we focus on (34).

To this end, recall the following result from [6], we rephrase it slightly differently for convenience:

Theorem 3 ([6]). *Assume the following condition for the respective periods of the writing/bus/reading systems:*

$$\lambda^{\mathbf{w}} \geq \lambda^{\mathbf{b}} \quad , \textit{and} \quad \left\lfloor \frac{\lambda^{\mathbf{w}}}{\lambda^{\mathbf{b}}} \right\rfloor \geq \frac{\lambda^{\mathbf{r}}}{\lambda^{\mathbf{b}}} , \qquad (35)$$

where, for x a real, $\lfloor x \rfloor$ denotes the largest integer $\leq x$. Then, the reader misses no data sent by the writer. Formally, there exists a strictly increasing sequence $k_n, n = 1, 2, \ldots$ of integers such that, for each n: $x^{\mathbf{r}}_{k_n} = x^{\mathbf{w}}_n$ and $\forall k : k_n \leq k < k_{n+1} \Rightarrow x^{\mathbf{r}}_k = x^{\mathbf{r}}_{k_n}$.

(In fact, a stronger result is proved in [6], allowing for slight drifts and jitter with respect to strict periodicity.) Now, the problem of "excessive sampling" at the reader can be compensated for by associating a boolean alternating flag to the data sent, so that switching of this flag marks, to the receiver, the successive instants k_n where correct sampling should occur. Note that the k_n sequence is not periodic in general. The original presentation of the protocol in [6] involved this flag from the beginning. We show here that its very reason is to enforce condition (34).

6. CONCLUSION

We developed a compositional theory of heterogeneous reactive systems. Logical time, physical time of various kinds, causalities, scheduling constraints, the simple local ordering of events of each individual signal as well as their combination, can be captured by our approach. We also developed a behavioural theory of heterogeneous architectures. We use it to formally study the process of deployment. This theory is rich enough to support general theorems about the correctness of deployment. This framework makes it relatively straightforward to study formally the correctness of the design principles in use at Airbus, based on the Ltta architecture.

Our models are denotational; they deal only with "traces", not with "agents" or "machines". Therefore, their usefulness in an automated design flow is questionable. Their value is mostly in providing a mathematical machinery to prove theorems about the correctness of particular methods and to develop solid foundations to design. We are about to extend the approach to an agent-based framework so that tools could be developed effectively to generate correct deployments.

7. REFERENCES

[1] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. J. Pappas and O. Sokolsky. Hierarchical Modeling and Analysis of Embedded Systems. *Proc. of the IEEE,* 91(1), 11–28, Jan. 2003.

[2] A. Benveniste, B. Caillaud, and P. Le Guernic. From synchrony to asynchrony. In J.C.M. Baeten and S. Mauw, Eds., *CONCUR'99, Concurrency Theory, 10th Intl. Conference*, LNCS 1664, pages 162–177. Springer, 1999.

[3] A. Benveniste, B. Caillaud, and P. Le Guernic. Compositionality in dataflow synchronous languages: specification & distributed code generation. *Information and Computation,* 163, 125-171 (2000).

[4] A. Benveniste, B. Caillaud, L. P. Carloni, P. Caspi, and A. L. Sangiovanni-Vincentelli. Composing Heterogeneous Reactive Systems. Submitted to ACM Transactions in Embedded Computing Systems.

[5] A. Benveniste, L. P. Carloni, P. Caspi, and A. L. Sangiovanni-Vincentelli. Heterogeneous reactive systems modeling and correct-by-construction deployment. In R. Alur and I. Lee, Eds., *Proc. of the 3rd. Intl. Conf. on Embedded Software, EMSOFT'03*, LNCS 2855, Springer, 2003.

[6] A. Benveniste, P. Caspi, P. Le Guernic, H. Marchand, J-P. Talpin and S. Tripakis. A Protocol for Loosely Time-Triggered Architectures. In A. Sangiovanni-Vincentelli and J. Sifakis Eds., *Proc. of the 2nd Intl. Workshop, EMSOFT 2002*, LNCS vol. 2491, 252-265, Springer, 2002.

[7] G. Buttazzo. Scalable Applications for Energy-Aware Processors. In A. Sangiovanni-Vincentelli and J. Sifakis Eds., *Proc. of the 2nd Intl. Workshop, EMSOFT'02*, LNCS vol. 2491, 153-165, Springer, 2002.

[8] P. Caspi. Embedded control: from asynchrony to synchrony and back. In T.A. Henzinger and C.M. Kirsch Eds., *Proc. of 1st Int. Workshop on Embedded Software, EMSOFT'01,*, LNCS 2211, 80–96, Springer, 2001.

[9] J. Eker, J.W. Janneck, E.A. Lee, J. Liu, J. Ludwig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity—The Ptolemy approach. *Proc. of the IEEE,* 91(1), 127–144, Jan. 2003.

[10] L. de Alfaro and T.A. Henzinger. Interface Theories for Component-Based Design. In T.A. Henzinger and C.M. Kirsch Eds., *Proc. of 1st Int. Workshop on Embedded Software, EMSOFT'01,* LNCS 2211, Springer, 2001.

[11] R. Goldblatt. *Topoi, the categorical analysis of logic.* Studies in logic and the foundations of mathematics, Vol. 98, North-Holland, 1984.

[12] E.A. Lee and Y. Xiong. System-Level Types for Component-Based Design. In T.A. Henzinger and C.M. Kirsch Eds., *Proc. of 1st Int. Workshop on Embedded Software, EMSOFT'01,*, LNCS 2211, Springer, 2001.

[13] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty. Model-Integrated Development of Embedded Software. *Proc. of the IEEE,* 91(1), 127–144, Jan. 2003.

[14] H. Kopetz. Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers. 1997. ISBN 0-7923-9894-7.

[15] E.A. Lee and A. Sangiovanni-Vincentelli. A Framework for Comparing Models of Computation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* 17(12), 1217–1229, Dec. 1998.

[16] D. Potop-Butucaru, B. Caillaud and A. Benveniste. Concurrency in Synchronous Systems. In *Proc. of the 4th Int. Conf. on Applications of Concurrency in System Design (ACSD)*, Hamilton, Canada, June 2004.