Actors without Directors: a Kahnian View of Heterogeneous Systems*

P. Caspi[1], A. Benveniste[2], R. Lublinerman[3], and S. Tripakis[1]

[1] Verimag, Grenoble, France
[2] INRIA/IRISA, Rennes, France
[3] Pennsylvania State University

Abstract. This paper aims to simplify recent efforts proposed by the Berkeley school in giving a formal semantics to the Ptolemy toolbox. We achieve this by developing a simple and elegant *functional* theory of deterministic tag systems that is a generalisation of Kahn Process Network theory (KPN). Our theory extends KPN by encompassing networks of processes labelled by tags from *partially ordered* sets and makes deeper use of Scott theory of Complete Partial Orders (CPO). Since CPO compose well under direct sums, heterogeneous systems are simply captured by *direct sums of homogeneous systems*, which are in turn constructed by connecting systems over different tag sets by means of *tag conversion* processes. For the (large) class of tag systems of "stream" type, we show how to define tag conversion processes and how to implement process communication. The resulting architecture is fully decentralised and does not require Ptolemy's directors. Last but not least, it provides distribution for free.

1 Introduction

The semantics of heterogeneity. The need for heterogeneity in modelling and development tools has been increasing while applications are becoming more and more complex. In view of this state of matters, pioneering frameworks like Ptolemy [11,13] which have started addressing the issue of heterogeneity a long time ago are becoming always more popular and raising an ever growing interest. Thus, concepts of this framework like *models of computation and communication* (MoCC), *actors*, *directors*, and so on, have been getting an increasingly larger acceptance.

Among the problems raised by this subject, the semantic question is important. While homogeneous applications are in general well-mastered, problems start at their interfaces, when several subsystems are composed to form a larger application. Ambiguities, semantic inconsistencies, etc., are likely to produce undesired behaviours which can badly impair the overall functioning of the composed application. To this end, Lee & Sangiovanni have introduced their celebrated tagged signal model [10] which was meant to provide a precise semantics to such frameworks as Ptolemy.

* This work was funded in part by the European Artist-Design Network of Excellence and the European STREP-COMBEST project number 215543

Yet, there was still a large gap between this denotational formalism and the behaviour of Ptolemy which is still largely bound to the operational semantics of the simulation engine. Efforts have been devoted to filling this gap: for instance BIP which is based on operational semantics [2] and 42 [14] which provides building blocks for designing MoCCs in a comprehensive way.

The application of Scott theory to tag systems. A comprehensive step toward closing this gap has recently been taken in [11], so as to make things simpler by getting rid of non-determinism, that is, by restricting from relations to functions. After all, determinism is something designers are fond of, most simulators like Ptolemy are deterministic and, when non-determinism is needed, in most cases it can be emulated by adding extra inputs to functions, aiming at choosing between several possible futures.[4]

Yet, this was not sufficient: when composing functions, inputs of one function can become outputs of another one and conversely, creating feedback loops and resulting in the functional aspect being lost: we get systems of equations which can have no solution as well as several solutions.

But this is a well-known issue of denotational semantics and well-known solutions exist. The most widely adopted one is Scott's semantics [15]: if the domain of interest is a complete partial order (CPO) and we restrict ourselves to continuous functions, then we know that every system of equations has a least solution and it is sensible to decide that this is the semantics of the system. Moreover, the least solution is itself a continuous function of its free inputs and thus can in turn be composed at will. The framework is thus closed by composition (and even by lifting to higher orders) and works perfectly well.

But there was another problem. The basic objects of the tagged signal model are signals which in a deterministic point of view can be seen as functions from tags to values. Scott approaches turn these signals into CPOs by turning the value set into a CPO. In this way, the CPO property gets automatically lifted from the image set to the function set. Thus, in this Scott theory applied to tag systems, the tag set does not need to have any order property. But, in tag system theory, tag sets are partially ordered and have a strong time flavour: in Ptolemy, computations go from past to future, while in the Scott framework, it does not matter (tags may not have any order and there may be neither past nor future!).

Towards Kahn semantics. Thus [13] had to modify Scott's order by requiring a prefix ordering principle in the spirit of the Kahn order [9]: a signal is larger than another one not only if it is more defined but also if both signals are defined on some initial segment of the tag set. In this way, a signal s_1 is larger than another signal s_2 if the initial segment over which s_1 is defined is larger than the initial segment over which s_2 is defined. In this way, computations can only extend the initial segments on which signals are defined and naturally flow from past to future.

[4] This is the way probability theory works: by adding input spaces about which the only knowledge we can have is their probability measure.

There was still a problem due to the fact that some tag sets, for instance associated with the discrete event (DE) domain, are infinite in several dimensions: in this case, initial segments are infinite and thus a signal defined over an initial segment has to have an infinity of values. In some sense, time may not progress, as in the so-called *Zeno* phenomenon of timed systems. But it is not possible to compute an infinite number of values in a simulator. In [13,11] the problem is solved using the idea of *absent value* from the French synchronous language school [3]: thus a signal defined on an initial segment may have only a finite number of computed non-absent values (while absent values need not be computed).

Paper's objectives and organisation. In this paper we develop a simple and elegant *functional* theory of deterministic tag systems that is a generalisation of Kahn's theory of Process Networks (KPN); KPN theory is recalled in section 2. As developed in section 3, our theory extends KPN by encompassing networks of processes labelled by tags from *partially ordered* sets and makes deeper use of Scott theory of Complete Partial Orders (CPO); since CPO compose well under direct sums, heterogeneous systems are simply captured by *direct sums of homogeneous systems*, which are in turn constructed by connecting systems over different tag sets by means of *tag conversion* processes. For the (large) class of tag systems of *stream* type introduced in section 4, we show how to implement process communication and how to define tag conversion processes (see section 6). Examples of tag systems are provided in section 5. Finally, we show in section 6 that the resulting architecture: 1) is fully decentralised; 2) does not require Ptolemy's directors, and 3) provides distribution for free. An extended presentation of this work can be found in [1].

2 Background on Deterministic Tag Systems and Kahn theory

Signals, Deterministic Signals, and Processes. The basic idea of the Tagged Signal Model [10] is to consider a signal $x \in S$ as a set of *events,* consisting of a pair "(tag, value)". Signals can thus be formalised as: $S = \{s \mid s \subseteq \mathbb{T} \times V\}$, where V is a set of values, and $(\mathbb{T}, \leq)$ is a partially ordered set of tags. These signals are non-deterministic ones: several values can be associated with the same tag. As we aim at considering deterministic tag systems, we first need to consider deterministic signals. This amounts to saying that we only consider signals that are partial functions from tags to values which we denote as: $S = \mathbb{T} \hookrightarrow V$.

In the deterministic setting, processes (or actors, following the Ptolemy terminology) are just functions transforming input signals into output signals. For the sake of simplicity, we do not consider the types of signal values and assume an "universal" type V. Thus, the set of processes, P, is just the set of total functions from S^m to S^n: $P = S^m \mapsto S^n$, where m, n are the input and output arities.

Functional Composition and Feedback Loops. In this deterministic setting, things are very simple. Processes are composed by functional composition and a composed process is just a system of equations, *e.g.,*

$$x_3 = p_1(x_1, x_2) \qquad x_4 = p_2(x_1, x_3)$$

which can define another process p_3 such that $(x_3, x_4) = p_3(x_1, x_2)$. However, this raises the question of feedback loops: for instance consider the system:

$$x_3 = p_1(x_1, x_2) \qquad x_2 = p_2(x_1, x_3)$$

What does it compute? This system of equations may have no solution or it may have several solutions. Then determinism can be lost.

Scott Semantics. Scott semantics [15] provides a well-known solution to this issue. It consists of the following changes to the previous framework:

1. Add to V an undefined element $\perp$ and a partial order relation $\leq$ such that:
   - $V^\perp = V \cup \{\perp\}$
   - $\leq$ is the least order relation over $V^\perp$ generated by: $\forall v \in V, \perp \leq v$.

   This makes $(V^\perp, \leq, \perp)$ a (flat) CPO. $\perp$ is the least element of $V^\perp$ and any sequence of ordered elements (a chain) has a least upper bound ($\bigvee$) which is $\perp$ if the chain contains only $\perp$'s, or some v_1 if the chain contains this v_1: note that in the latter case the chain cannot contain another v_2 distinct from v_1 as the two are incomparable.
2. Redefine S as the set of *total* functions from $\mathbb{T}$ to $V^\perp$:[5] $S = \mathbb{T} \mapsto V^\perp$. Then S inherits the CPO property of $V^\perp$ by defining:
   - $x \leq x'$ if for all $t \in \mathbb{T}$, $x(t) \leq x'(t)$ which amounts to saying that x is smaller than x' if it is less defined,
   - the bottom element of S, also denoted $\perp$, as the signal which is undefined everywhere: $\perp(t) = \perp$.

   Given a chain of signals $x_0, \ldots x_n, \ldots$, and given any tag t, $x_0(t), \ldots x_n(t), \ldots$ is chain of values and

$$\bigvee\{x_0, \ldots x_n, \ldots\}(t) = \bigvee\{x_0(t), \ldots x_n(t), \ldots\}$$

3. Restrict processes to *continuous* functions from input to output signals, which means that, given a chain of inputs, that is to say a sequence of more and more defined signals, the outputs should form a chain and

$$\bigvee\{p(x_0), \ldots p(x_n), \ldots\} = p(\bigvee\{x_0, \ldots x_n, \ldots\}) \tag{1}$$

   Note that continuity implies *order preservation*: $s \leq s' \Rightarrow p(s) \leq p(s')$ and note that this definition for single-input/single-output processes can be extended naturally to processes of different arities because products of CPOs inherit the CPO structure of their components. In particular, the order on the product is the component-wise order.

[5] A partial function can be made total by giving it the value $\perp$ whenever it is not defined.

4. Then the Kleene theorem says that *any system of equations has a (unique) least solution, which is in turn a continuous function of its free input signals.* Thus composition preserves determinism and confluence of unscheduled distributed executions is guaranteed.

Kahn Theory. But this solution is still unsatisfactory because it does not take advantage of the ordering over tags which have a flavour of time. In particular, a process may as well compute from future to past—we can easily design a process that is continuous in the Scott sense but not causal. This issue of causality is properly addressed by Kahn theory.

Kahn's world is a special case of Scott's world. In Kahn's world, the tag domain is $\mathbb{N}$, which is a totally-ordered and enumerable set. Signals are partial functions from $\mathbb{N}$ to a set of values V. In addition, all signals are assumed to be *prefix-closed*, meaning that if they are undefined at some time n then they remain undefined for all $n' > n$.

Note that in Kahn's original paper [9] signals are elements of $V^\infty = V^* \cup V^\omega$ where V^* is the set of all finite sequences over V and V^ω is the set of all infinite sequences over V. The set of all prefix-closed signals from $\mathbb{N}$ to V is isomorphic to V^∞: partially-defined signals correspond to finite sequences and totally-defined signals to infinite sequences.

Looking at signals x and y as sequences, $x \leq y$ means that x is a prefix of y (there exists a sequence x' such that $y = x \cdot x'$, where $x \cdot x'$ denotes the concatenation of x and x'). With this order, the set of all signals becomes a poset. It is in fact a CPO: (1) $\bot$ is the empty sequence ϵ; (2) the least upper bound of a chain of increasingly defined signals is either: (2.1) the most defined one if the chain is finite or: (2.2) the infinite sequence defined by the chain, because an infinite sequence is a maximal element of the CPO (concatenating a sequence to an infinite sequence does not change this sequence).

Processes are assumed to be continuous functions from S^m to S^n. Again we can define the semantics as the least solution of systems of equations. Thus the Kahn theory solves the feedback loop problem. But it also solves the causality problem: a continuous process is order preserving and, in terms of Kahn order, this means that if x is a prefix of y, it is also the case that $f(x)$ is a prefix of $f(y)$. This means that the future of x cannot influence the present of x. Computations are guaranteed to flow from past to future.

3 Kahn generalisation to Partially Ordered Tag sets

Kahn network over a Partially Ordered Tag set. Kahn signals can be seen as tagged signals over the particular tag set $\mathbb{N}$. In order to address heterogeneity, we would like to generalise Kahn's approach to other tag sets. The technical difficulty here is that in general, tag signals cannot be prefix-closed because prefixes can be infinite. While in [13,11] this problem is solved by introducing a special "absent" value, we propose here an alternative, more general approach, that does not require absent values and is still compatible with infinite prefixes.

Let S be the set of all total functions

$$S = \mathbb{T} \mapsto V^{\perp}, \tag{2}$$

where $\mathbb{T}$ is a poset. Following Kahn, we endow S with the following partial order:

Definition 1 (Prefix order over signals) *A signal x is a prefix of a signal y, if for all t, $y(t) \neq x(t)$ implies for all $t' \geq t$, $x(t') = \perp$.*

It is easy to see that this is indeed an order relation. Please note also that this definition allows "holes" in the defined values; for instance we could have:

$$x : 1, \perp, 2, \perp, \perp, \perp \ldots \qquad y : 1, \perp, 2, \perp, 3, \perp \ldots$$

In the above example we have indeed $x \leq y$. Note that in this example $\mathbb{T} = \mathbb{N}$.

Proposition 1 (CPO) *S endowed with the prefix order is a CPO.*

Proof. First, take $\perp(t) = \perp$. Then we notice that given $x \leq y$ in S, for any t, $x(t) \leq y(t)$ according to the CPO $V^{\perp}$. Thus, if $\{x_n\}$ is a chain, then, for any t, $\{x_n(t)\}$ is a chain and we can take: $\bigvee\{x_n\}(t) = \bigvee\{x_n(t)\}$.

Definition 2 *A process p is* order-preserving *if, for any two signals, x, y if x is a prefix of y, then $p(x)$ is a prefix of $p(y)$; p is* continuous *if it satisfies (1).*

This means that only the past can influence the present value of a process. The mathematical framework of this section is both simple and very powerful. Restricting to order-preserving processes allows us to preserve *determinism, causality,* and *confluence* of unscheduled distributed executions.

Capturing Heterogeneity via Sum of CPOs. The next problem is to extend the previous generalised Kahn theory to encompass heterogeneity, that is, systems involving different tag sets. But this in our framework comes for free: The *sum* of two CPOs S_1 and S_2 is a CPO $S_1 + S_2$, defined as

$$S_1 + S_2 = (S_1 - \{\perp_{S_1}\}) \cup (S_2 - \{\perp_{S_2}\}) \cup \{\perp\}$$

where $\perp, \perp_{S_1}$, and $\perp_{S_2}$ are the corresponding bottom elements. The order on each of S_1, S_2 is maintained in the sum, but two elements from two different sets are not comparable. Therefore, chains can only be formed of elements of a single set and the least upper bounds are preserved.

Heterogenous architectures. At this point, suppose that we know how to construct *tag conversion functions,* i.e., continuous functions

$$f : S_1 \to S_2. \tag{3}$$

Let us see now how such a function can be seen as operating on the sum CPO $S_1 + S_2$. Indeed, $f : S_1 \to S_2$ can be seen as a function $f' : (S_1 + S_2) \to (S_1 + S_2)$ by setting:

$$f'(\bot) = \bot, \qquad f'(in_1(x)) = f(x), \qquad f'(in_2(x)) = \bot$$

where in_1, in_2 are the canonical injection of each CPO into the sum. Next, consider the following toolkit of functions, consisting of:

- homogeneous functions, mapping input signals to output signals belonging to a domain S of signals over a same partially ordered tag set $\mathbb{T}$;
- tag conversion functions, mapping an input signal over $\mathbb{T}_1$ to an output signal over $\mathbb{T}_2$.

By using the previous reasoning, a finite network of such functions can be seen as a network of homogeneous functions acting on the direct sum $\sum_{i \in I} S_i$, where finite set I indexes the set of homogeneous functions of the considered network. By proposition 1, the network itself is an homogeneous function acting on the direct sum $\sum_{i \in I} S_i$. Thus this network itself can be encapsulated as a function acting on $S =_{\text{def}} \sum_{i \in I} S_i$, so the same construction can be reused, hierarchically. Observe that we can also encapsulate tuples of signals over different tag sets as a single signal defined over the sum of the considered tag sets. In other words, hierarchy can be used for both boxes (functions) and wires (signals). Having this architecture model addresses the main objective of this paper.

The remaining problems. From the previous analysis, the following two central issues remain to be addressed:

Problem 1. How to construct tag conversion functions?

Having a solution to problem 1 provides us immediately with a framework of heterogeneous Kahn-like architectures, as explained just above.

Problem 2. How to implement wires carrying signals defined over a partially ordered tag set $\mathbb{T}$?

This problem also remains to be solved in order to make our approach effective—recall that such an implementation exists for basic Kahn networks since the latter rely on a communication medium of unbounded FIFOs. Other media may be needed for other partially ordered tag sets. We address Problem 1 in Section 4 and Problem 2 in Section 6 and show how to solve them in the restricted case of "streams".

4 Streams: from Generalised to Ordinary Kahn Theory

In the generalised Kahn theory developed in section 3, signals are total functions from a partially ordered tag set $\mathbb{T}$ to a set of values $V^\bot$, or, equivalently, partial functions from $\mathbb{T}$ to V. These signals can thus be seen as "labelled partial orders"

and are fairly general. Yet, in many practical cases, for instance those which correspond to what is considered in Ptolemy [13] and which are addressed in section 5, this generality is not needed and the approach can be simplified. This simplification is based on two assumptions. When these assumptions are in force, signals can be seen as streams and the theory boils down to an ordinary Kahn theory. While this reduction is unnecessary from a mathematical standpoint, it has practical applicability, since we know that *Kahn networks* (unlike general tag systems of section 3) *are implementable on networks of processors related by FIFO links, cf. our problem 2.*

Assumption 1 (DTOS) *In the considered set S of signals:*

1. *The tag set is a total order.*
2. *The defined values of any signal can be indexed in non-decreasing order, meaning that there is an order preserving isomorphism from the domain of any signal, $dom(x) = \{t \mid x(t) \neq \bot\}$, to (an initial segment of) $\mathbb{N}$.*

Call Discrete over Totally Ordered tag Set (DTOS) *such a set S of signals.*

This means that there is an order preserving isomorphism from the domain of a signal ($dom(x) = \{t \mid x(t) \neq \bot\}$) to (an initial segment of) $\mathbb{N}$. We call this a *discrete signal*. Assumption 1 yields signals whose defined tags are order isomorphic to (an initial segment of) $\mathbb{N}$.[6] In this case we speak of a *discrete total order*.

An important question arising from Assumption 1 is whether the restriction of signals based on these assumptions still preserves the CPO structure defined in proposition 1. This is by no means a trivial issue. The following proposition provides a positive answer.

Proposition 2 (DTOS) *The set of* Discrete signals over a Totally Ordered tag set (DTOS) *endowed with the prefix order is indeed a CPO.*

The proof can be found in [1].

DTOS signals as streams. When dealing with DTOS signals, tags associated to defined values can be indexed in increasing order and signals can be seen as streams of pairs (value, tag). In other words, to a DTOS signal x we can associate its *stream* $St(x) : (V \times \mathbb{T})^\infty$ where there is no more need to consider an undefined value.[7] The tag ordering constraint is then for any $s \in DTOS$ such that $s = (v_1, t_1).(v_2, t_2).s'$,

1. $t_1 < t_2$
2. $s' \in$ DTOS

It is then clear that the stream view of DTOS signals enjoys the same properties as the functional one. Formally the DTOS-to-stream transformation is as follows:

$$\begin{aligned} St(\bot) &= \epsilon, \text{ the empty sequence} \\ St(x) &= (x(t_1), t_1).St(x[t_1 \rightarrow \bot]) \end{aligned} \tag{4}$$

[6] Note the importance of requiring an order-preserving isomorphism in Assumption 1. Rationals are both totally ordered and countable but not order isomorphic to $\mathbb{N}$.

[7] This view is inspired by our previous work [4].

where:

- t_1 is the least tag yielding a defined value in x
- $x[t_1 \to \bot]$ is the function x where the value at t_1 has been changed to $\bot$.

To conclude this section, we formally state the following property:

Proposition 3 *St defined in (4) is a CPO isomorphism between* DTOS *signals and streams. Moreover, it preserves parallel composition. (This solves problem 2.)*

Thus, DTOS systems can be brought back to streams, i.e., ordinary Kahn networks.

5 Examples

Kahn Process Networks. Kahn Process Networks (KPN) naturally fit into that landscape. $\mathbb{N}$ is the tag set and there are no "holes" between defined values. Thus signals are just streams of defined values. An operator like the sum operator over numbers can be lifted to streams according to the following Haskell-like definition:

$$\begin{aligned} \text{sum}_K\ \epsilon\ y &= \text{sum}_K\ x\ \epsilon ;=\ \epsilon \\ \text{sum}_K\ v.x\ v'.y &= (v+v').\text{sum}_K\ x\ y \end{aligned}$$

where ϵ denotes the empty stream and "." denotes concatenation. Basically, the Kahn actor sum_K waits until both its input queues are non-empty. Then, it removes their heads, adds them, puts the result into its output queue and starts again. Note that waiting until both queues are non-empty can be implemented using Kahn's *blocking read* operator, without having to test both queues *simultaneously*: sum_K simply blocks on one queue, then on the other. The order in which queues are read can be arbitrary.

Discrete Event. We begin with a tagged view of Discrete Event Signals and then present a streamed view for them.

A Tagged View of Discrete Event (DE) Signals. Discrete event (DE) signals are discrete signals (according to assumption 1) with real-time stamps. A tag set for DE is:

$$\mathbb{T} = \mathbb{R}_+ \times \mathbb{N}$$

where $\tau = (t, n)$, t denotes a time stamp, and n is the index of events sharing the same time stamp.[8] This tag set is ordered with the lexicographic order:

$$(t, n) \leq (t', n') \text{ iff either } t < t' \text{ or } t = t' \text{ and } n \leq n'$$

which is a total order. Then a discrete event signal x is a total function $x : \mathbb{T} \mapsto V^\bot$ satisfying the following constraint: for any two tags $\tau, \tau' \in \mathbb{T}$, with $\tau = (t, n)$, $\tau' = (t, n')$, and $n \leq n'$, if $x(\tau') \neq \bot$ then $x(\tau) \neq \bot$. Figure 1 shows an example of such a signal. In this example the following table provides the correspondence between tags and values:

[8] This approach, which has been called *super-dense time* by some authors [13], could be easily extended to tag sets $\mathbb{T} = \mathbb{R}_+ \times \mathbb{N}^{\mathbb{N}}$ to account for so-called "nested oversamplings". For the sake of simplicity, we do not address such an extension here.

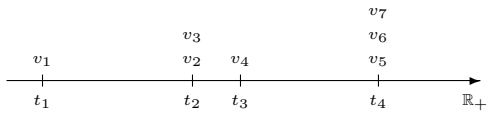

Fig. 1. A discrete-event signal.

$$\begin{array}{rccccccc} \text{tag}: & (t_1,1) & (t_2,1) & (t_2,2) & (t_3,1) & (t_4,1) & (t_4,2) & (t_4,3) \\ \text{value}: & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array}$$

We can remark that in this definition, there are many undefined (or absent) values namely between two consecutive time stamps holding defined values and after the last defined value sharing a given time stamp.

A Streamed View of Discrete Event Signals. DE signals are DTOS, so we can apply the results of section 4, thus providing a streamed view of them:

$$sx : (V \times (\mathbb{R}_+ \times \mathbb{N}))^\infty$$

where $\mathbb{R}_+$ is the set of non-negative reals modelling the physical (or *real*) time. Furthermore we observe that in this definition, the second component $\mathbb{N}$ of the tag set is not necessary because we can always rebuild it by applying the following index rebuilding mapping $\text{Ir} : (V \times \mathbb{R}_+)^\infty \to (V \times (\mathbb{R}_+ \times \mathbb{N}))^\infty$:

$$\begin{array}{ll} \text{Ir}_1(t', n, \epsilon) & = \epsilon \\ \text{Ir}_1(t', n, (v,t).sx) & = \text{if } t == t' \\ & \quad \text{then } (v,(t,n+1)).\text{Ir}_1(t, n+1, sx) \\ & \quad \text{else } (v,(t,1)).\text{Ir}_1(t,1,sx) \\ \text{Ir}(\epsilon) & = \epsilon \\ \text{Ir}((v,t).sx) & = (v,(t,1)).\text{Ir}_1(t,1,sx) \end{array}$$

An Actor Example It is interesting to see how to define some primitive actors in DE. Let us start by defining the sum of two signals. There are several ways of defining it, each having, perhaps surprisingly, very different properties [1]. Here we present only one possibility, which states that, when both input signals appear with the same tag, we output the sum, otherwise we just output the defined signal:

$$\begin{array}{ll} \text{sum}_{DE2}\ x\ \epsilon & = \text{sum}_{DE2}\ \epsilon\ y\ =\ \epsilon \\ \text{sum}_{DE2}\ (v,\tau).x\ (v',\tau').y & = \text{if } \tau < \tau' \\ & \quad \text{then } (v,\tau).\text{sum}_{DE2}\ x\ (v',\tau').y \\ & \quad \text{else if } \tau = \tau' \\ & \qquad \text{then } (v+v',\tau).\text{sum}_{DE2}\ x\ y \\ & \qquad \text{else } (v',\tau').\text{sum}_{DE2}\ (v,\tau).x\ y \end{array}$$

Figure 2 illustrates this definition which can be proved to be continuous ([1]) though it uses, unlike in KPN, the infamous[9] operation that tests values in input queues without removing (consuming) them.

[9] because its undisciplined usage may result in non-continuous processes

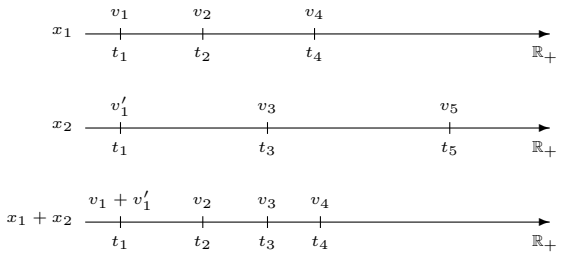

Fig. 2. Sum (DE2) of two discrete-event signals.

Other operators can be found in [1].

Continuous Time. In general, the continuous time (CT) case is more involved, and its study is part of our on-going work. Some preliminary ideas can be found in [1]. We summarise these here.

First, note that there are different CT domains, depending on whether we want to define signals with *exact* (i.e., ideal) CT semantics, or *approximate* CT semantics, as computed using a numerical solver. Exact semantics is linked to the theory of ordinary differential equations (ODEs) (see also [12] and [6]). Yet it seems to us that exact CT does not fit into the Kahn landscape: in order to exactly solve a differential equation, this equation has to be considered globally as a whole and it cannot be decomposed into its components. For instance we cannot define the exact behaviour of an integrator, independently from the network of operators which feeds it. Indeed, in theory we would need to check whether this network computes a Lipschitz function or not.

Regarding approximate semantics, observe that ODEs, when discretised using explicit schemes with fixed step size, are simply DE systems, thus can be handled in the DE domain. However, this no longer holds if more sophisticated schemes are used, e.g. implicit schemes and/or variable step size.

Synchronous Reactive. Synchronous Reactive systems have been addressed among others in [5,7]. It is in this domain that absent values have been first introduced. Here also, we begin with a *Tagged view of Synchronous Reactive* (SR) systems. These are very similar to discrete event ones, but real-time is replaced with a logical integer time. Thus the tag set is[10] $\mathbb{N} \times \mathbb{N}$ where the first component gives the reaction logical time and the second one the multiplicity index in that reaction. The reason we need both components is to model easily *multi-clock* systems: in such systems, a signal may be "absent" in some reactions (captured by $\perp$) and occur several times in other reactions. In some sense, the reaction logical time acts as a replacement of absent values.

[10] The same remark on a possible extension to nested oversamplings as stated at section 5 applies here.

For the *Streamed view of Synchronous Reactive*, we can proceed as with DE signals: $St(x) : (V \times \mathbb{N})^{\infty}$. In this interpretation $St(x)(n) = (v, r)$ means the n-th occurrence of x has value v and takes place within the r-th reaction.

An example that is not DTOS. So far all examples we have presented are DTOS. It will not be the case for the following one, however. The MoCC we present here is that of signals that are themselves streams of events, however with causality relations between events belonging to different signals. To formalise this example we need an underlying set X of signal *names*. The set $\mathbb{T}$ of tags has the following form, where $\mathbb{N}_{\infty} = \mathbb{N} \cup \{-\infty\}$:

$$\mathbb{T} = X \times (X \to \mathbb{N}_{\infty})$$

In other words, $t \in \mathbb{T}$ has the form $t = (x, \tau)$, where τ is a *vector clock*, i.e., a total function mapping X to $\mathbb{N}_{\infty}$. The interpretation of $t = (x, \tau)$ is as follows:

- tag t belongs to a signal with name x;
- signal x is indexed by the set of positive integers $\mathbb{N}$ and its rank is given by $n = \tau(x)$, which must therefore be $> -\infty$;
- a positive value for $\tau(y) = m > -\infty$, where $y \in X \setminus \{x\}$ indicates a causality constraint of the nth event of signal x with respect to the mth event of signal y; having $\tau(y) = -\infty$ indicates lack of causality constraint of the nth event of signal x with respect to any event of y.

We may (but do not need to) restrict $\mathbb{T}$ to tags whose vector clock τ takes a value $\tau(y) \neq -\infty$ for only finitely many y's. $\mathbb{T}$ is equipped with the following order relation, making it a partial order:

$$t \geq t' \text{ iff } \forall y \in X \text{ s.t. } \tau'(y) > -\infty \Rightarrow \tau(y) \geq \tau'(y)$$

6 Actors without Directors

6.1 Tag Conversion Actors in lieu of Directors

When dealing with heterogeneity, there is generally no "golden rule" saying what the meaning of composing actors with different tag sets should be. This information must instead be provided by the designer.

In Ptolemy this problem is solved using the concept of *directors*. Roughly speaking, a director schedules the operation of a set of concurrent actors in time, thus in essence defining the concurrency (and time) semantics of the model. There are many types of directors in Ptolemy, each implementing a given MoCC: discrete-event, synchronous-reactive, etc.

Here, we take a different approach: we define *tag-conversion* actors, *i.e.,* heterogeneous actors operating on different tag sets and transforming signals on one tag set to signals on another tag set. Compared to directors, our approach has two main advantages. First, we do not need to introduce an additional concept in our modelling framework, actors is all we need. Second, our approach allows to separate the issue of semantic compatibility from that of using different MoCCs.

We give now some standard conversion actors to allow the interconnection of signals of different tags. These are only a few examples and other tag-conversion actors can obviously be defined.

From DE and SR to KPN. Going from DE and SR to KPN can be done by "forgetting" the tag:

$$\begin{array}{lll} \text{forget} & \epsilon & = \epsilon \\ \text{forget} & (x,t).xs & = x.\text{forget}\ \ xs \end{array}$$

From KPN to DE and SR. In the opposite direction, a "timestamping" actor can be used. This actor uses a clock that specifies the timestamps:

$$\begin{array}{lll} \text{timestamp} & \epsilon\ cl & = \epsilon \\ \text{timestamp} & xs\ \epsilon & = \epsilon \\ \text{timestamp} & (x.xs)\ (t.ts) & = (x,t).(\text{timestamp}\ xs\ ts) \end{array}$$

6.2 Distribution

A stated at the end of section 3, restricting to order-preserving processes in the sense of definition 2 allows us to preserve determinism, causality, and *confluence of unscheduled distributed executions*. Thus, distribution comes for free and does not need coordination. This holds in particular for heterogeneous models mentioned at the end of section 6.1.

Still, the following issue remains, namely: *which type of communication link is needed in such distributed implementations?* Since tag conversion is performed by actors, *links involve only homogeneous tag sets*. So, in general, our (directed) links only need to preserve the prefix order of definition 1 for a given (homogeneous) tag set $\mathbb{T}$, from source node to sink node. This holds in particular for heterogeneous models mentioned at the end of section 6.1. In particular, standard FIFO links can be used to implement communications for such architectures.

Take for instance the definition of the discrete event sum illustrated in figure 2 of section 5. The actor has two input FIFO queues `x` and `y` and an output FIFO queue `z`. The queues contain pairs $(v : V, t : \mathbb{R}_+)$. Indeed the $\mathbb{N}$ component of the DE tag set is useless because the FIFO queues preserve the order of production. Thus an operational version of the sum actor can be defined in C-like syntax as:

```
void sum(input queue x, input queue y, output queue z) {
  if (x.empty() OR y.empty()) return;
  if (x.head().tag() < y.head().tag()) {
     z.append(x.head()); x.erase_head(); return; }
  if (x.head().tag() == y.head().tag()) {
     z.append(x.head().tag(), x.head().val() + y.head().val());
     x.erase_head(); y.erase_head(); return; }
  // it must be that: x.head().tag() > y.head().tag()
  z.append(y.head()); y.erase_head(); return;
}
```

Basically, the sum process needs that its two input files be non-empty to execute. Otherwise it waits. If it can execute, it takes the two tagged heads and

compares their tags. If they are equal, it sums up the two values, tags the result with the common tag, puts it in the output queue and erases the two heads from the input queues. If the two tags are different, the earlier tagged value is erased from its input queue (as a matter of fact we know that, since the input queue values are produced in an orderly manner, it will not be possible that the other queue will later contain an item matching this earlier tag) and the other queue is left unchanged. Nothing is produced and the process waits.

Indeed, we could say, adopting the Ptolemy terminology that such networks do not need directors, *i.e.,* some *deus ex machina* able to schedule the executions of each actor. In a simulation engine, the only need is that execution is fairly distributed between actors in such a way that no actor is infinitely excluded from execution. Also note that there is no "event queue" like what is found in most simulation engines like Ptolemy and this feature avoids the burden of building a distributed event queue. Distributed actors are truly autonomous, they only know of the heads of their input queues.

6.3 Hierarchy

It has been advocated that the use of directors enforces a clean separation between several MoCCs in a hierarchical way: in order to get a communication between two different MoCCs these have to be encapsulated within a "larger" MoCC which encompasses the former ones. It is true that this is a good design practice but the "flat" directorless approach we present here is fully compatible with hierarchy: Kahn actors can be gathered so as to form compound actors and this hierarchical composition can be extended at will.

7 Conclusion

This paper has intended to simplify recent efforts proposed by the Berkeley school in giving a formal semantics to the Ptolemy toolbox. We have proposed a simple and elegant functional theory of deterministic tag systems that is a generalisation Kahn's theory of Process Networks (KPN). Our theory encompasses networks of processes labelled by tags from *partially ordered* sets and makes deeper use of Scott theory of Complete Partial Orders (CPO). Since CPO compose well under direct sums, heterogeneous systems are simply captured by *direct sums of homogeneous systems*, which are in turn constructed by connecting systems over different tag sets by means of *tag conversion* processes. For the (large) class of tag systems of *stream* or DTOS type, we have shown how to define tag conversion processes and how to implement process communication. The resulting architecture is fully decentralised and does not require Ptolemy's directors. Last but not least, it provides distribution for free.

A natural question is to find broader frameworks than just DTOS in which problems 1 and 2 can be properly solved. This is left for future work.

An important issue not addressed in the paper is the issue of liveness (also called productivity in the co-algebraic framework). This issue has already been

partially addressed in [13] and its adaptation to our stream approach will be a subject for future work.

Another semantic theory for stream-based systems, alternative to Kahn, is the co-algebraic theory of streams (see for instance [8]). Basically, moving to co-algebraic streams would consist, in the stream programs shown in the paper, to remove the ϵ cases. Indeed, this is what has been done in the Haskell prototype we have implemented of our framework. Examining the consequences of such an alternative choice is also a subject for future work.

References

1. Full version of this paper available as technical report TR-2008-6 from http://www-verimag.imag.fr/index.php?page=techrep-list.
2. A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-time Components in BIP. In *SEFM'06*, pages 3–12, 2006.
3. A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *IEEE Proceedings*, 79:1270–1282, September 1991.
4. A. Benveniste, B. Caillaud, L. P. Carloni, P. Caspi, and A. L. Sangiovanni-Vincentelli. Composing heterogeneous reactive systems. *ACM Trans. Embedded Comput. Syst.*, 7(4), 2008.
5. G. Berry and E. Sentovich. An implementation of constructive synchronous programs in polis. *Formal Methods in System Design*, 17:135–161, 2000.
6. S. Bliudze and D. Krob. Towards a functional formalism for modelling complex industrial systems. In *Complex Systems (ECCS'05)*, pages 163–176, 2005.
7. S. A. Edwards and E. A. Lee. The semantics and execution of a synchronous block-diagram language. *Science of Computer Programming*, 48(1), 2003.
8. B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of EATCS*, 62:229–259, 1997.
9. G. Kahn. The semantics of a simple language for parallel programming. In *IFIP*, 1974.
10. E. A. Lee and A. Sangiovanni-Vincentelli. A unified framework for comparing models of computation. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 17(12):1217–1229, December 1998.
11. E.A. Lee and H. Zheng. Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. In *EMSOFT'07*, 2007.
12. J. Liu and E.A. Lee. On the causality of mixed-signal and hybrid models. In *HSCC*, pages 328–342, 2003.
13. X. Liu and E.A. Lee. CPO Semantics of Timed Interactive Actor Networks. *Theoretical Computer Science*, 409(1):110–125, 2008.
14. F. Maraninchi and T. Bouhadiba. 42: Programmable models of computation for a component-based approach to heterogeneous embedded systems. In *GPCE*, 2007.
15. D. Scott. Data types as lattices. *SIAM J. on Computing*, 10(3):522–587, 1976.