# On the Verification of Timed Discrete-Event Models*

Christos Stergiou, Stavros Tripakis, Eleftherios Matsikoudis, Edward A. Lee

University of California, Berkeley

**Abstract.** Timed discrete-event (DE) is an actor-oriented formalism for modeling timed systems. A DE model is a network of actors consuming/producing timed events from/to a set of input/output channels. In this paper we study a basic DE model, called deterministic DE (DDE), where actors are simple constant-delay components, and two extensions of DDE: NDE, where actors are non-deterministic delays, and DETA, where actors are either deterministic delays or timed automata. We investigate verification questions on DE models and examine expressiveness relationships between the DE models and timed automata.

## 1 Introduction

Timed automata, introduced by Alur and Dill [2,7] are one of the prominent formalisms for timed systems. In this paper we study another formalism for modeling dense-time systems, which we call *timed discrete-event* (DE). DE is inspired by the DE domain of the tool Ptolemy [9], which is itself inspired by established fields such as discrete-event simulation and queueing models. More recently, DE has been proposed as a programming abstraction for cyber-physical systems and used in the Ptides framework [8] for the design of distributed real-time systems, e.g. control systems for electric power stations and correct shutdown mechanisms for industrial controllers.

DE is an *actor-oriented* formalism, in the sense of Agha [1], where a model consists of a collection of *actors*, each consuming and producing messages from input and to output channels. In the case of DE a model is a network of actors. Ptolemy DE models can be hierarchical, but for the purposes of this paper we only consider flat models. We also abstract away message values, thereby leaving actors to communicate only via *timed events*. A timed event is characterized by a single value, its timestamp.

* This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. It was also supported in part by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley (supported by NSF awards #0720882 (CSR-EHS: PRET) and #0931843 (ActionWebs), the Naval Research Laboratory (NRL #N0013-12-1-G015), and the companies: Bosch, National Instruments, and Toyota), and the NSF Expeditions in Computing project *ExCAPE: Expeditions in Computer Augmented Program Engineering*.

Even though discrete-event simulation is widespread in system design, including, for instance, hardware system simulation methods based on languages such as Verilog, VHDL, and SystemC, the exhaustive verification of DE models has received little attention, to our knowledge. In this paper we take a step towards remedying this, by studying three versions of DE models in terms of expressiveness and model-checking.

First, we introduce a basic, *deterministic DE* (DDE) model, where actors are simple constant (and known) delays. An actor in DDE delays every input event by a constant delay $\Delta$, which means that if the input event has timestamp $\tau$ then the actor produces a corresponding output event with timestamp $\tau + \Delta$. A constant delay actor cannot be represented by an equivalent timed automaton, as the latter would need an unbounded number of clocks, one for every input event that may arrive within an interval $\Delta$.

Nevertheless, we can show that the strong deterministic properties of DDE allow its state space to be reduced to a finite *lasso*. The latter can be used for exhaustive model-checking of both *signal* and *state* queries. An example of a signal query is "*is there an execution where an event with timestamp* $> 10$ *occurs in channel* $c$." An example of a state query is "*is there a reachable state where channels* $c_1$ *and* $c_2$ *contain two events with timestamps* $\tau_1, \tau_2$*, such that* $|\tau_1 - \tau_2| \leq 2$." The lasso can also be used to show that every DDE model can be transformed to an equivalent timed automaton (TA) model.

We also introduce two extensions to the basic DDE model: *non-deterministic DE* (NDE) and *DE with timed automata* (DETA). In NDE, actors are non-deterministic delays, specified by an interval, say, $[l, u]$, so that an input event is delayed by some arbitrary $\delta \in [l, u]$. In DETA, actors are either constant delays, or timed automata. A timed automaton $M$ can be viewed as an actor which reacts to input events arriving on a given channel $c$ by taking a discrete transition labeled with input $c$ (we require that $M$ be *receptive*, that is, always be able to accept any input). $M$ can spontaneously choose to generate an output event on a given channel $c'$ by taking a discrete transition labeled with $c'$.

Finally, we discuss expressiveness of the above models. We show that DDE $\subset$ NDE and DDE $\subset$ TA $\subset$ DETA, where all inclusions are strict. We also show that NDE $\not\subseteq$ TA, and conjecture that TA $\not\subseteq$ NDE and NDE $\not\subseteq$ DETA.

## 2 Deterministic timed discrete-event models

We abstract away event values, so events are only timestamped tokens. Formally, an event is represented by a timestamp $\tau \in \mathbb{R}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ is the set of non-negative reals. The set of naturals is denoted $\mathbb{N} = \{0, 1, 2, ...\}$.

### 2.1 Syntax

A DDE model is a finite labeled directed graph $G = (A, C, D)$ such that

- $A$ is the set of nodes of $G$. Each node is called a *DE actor* or *actor* in short.

– $C \subseteq A \times A$ is the set of edges of $G$. Each edge $c \in C$ is called a *channel*.
– $D : A \to \mathbb{N}$ is a (total) function mapping each actor $a \in A$ to a non-negative integer number called the *delay* of $a$. Note that $D(a)$ may be 0.

Let $c = (a, b) \in C$. Then $c$ is an *output* channel of $a$ and an *input* channel of $b$. We use $C^{in}(a)$ and $C^{out}(a)$ to denote the sets of input and output channels of an actor $a$, respectively. Let $C(a) = C^{in}(a) \cup C^{out}(a)$. By definition, $G$ is a *closed* model, in the sense that all input channels are connected. In fact, every channel has a unique writer and a unique reader. An actor without input (respectively, output) channels is called a *source* (respectively, *sink*).

An example of a DDE model is given Figure 1. The model has three actors, $a_1, a_2, a_3$, with delays $1, 1, 0$, respectively, and four channels (the four arrows).

A *channel state* for a DE model $G$ is a total function $r : C \to 2^{\mathbb{R}_{\geq 0}}$ which maps every channel $c \in C$ to a finite set of events initially pending on $c$. In Figure 1, the bullets annotating channels $c_1, c_2$ specify an *initial channel state*. In this case there are two initial events, both with timestamp 3.

*Partial order:* Our model allows cyclic graphs and zero-delay actors. However, we require that every cycle visits at least one actor $a$ such that $D(a) > 0$. This condition effectively allows to "break" zero-delay loops, and to define a partial order $\prec$ on the set of actors $A$, so that $a \prec a'$ iff there exists a path from $a$ to $a'$ such that for any actor $a''$ in the path (including $a$ but excluding $a'$) we have $D(a'') = 0$. The order $\prec$ is essential for ensuring that actors are fired in timestamp order (Lemma 1) which in turn yields important deterministic properties of the DDE model. For the example of Figure 1, the order $\prec$ is $a_3 \prec a_2$.

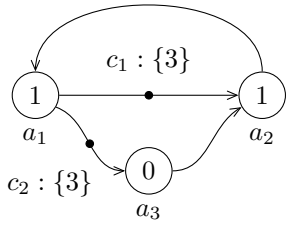



**Fig. 1.** A DDE model.

### 2.2 Operational Semantics

To a given DDE model $G$ and initial channel state $r_0$, we will associate a *timed transition system* $TTS(G, r_0) = (S, s_0, \to)$, where $S$ is its set of states, $s_0 = (r_0, 0)$ is its (unique) initial state, and $\to \subseteq S \times A \times S$ is its transition relation, defined below. A state $s \in S$ is a pair $(r, t)$, where $r$ is a channel state and $t \in \mathbb{R}_{\geq 0}$ is a *global timestamp*. The initial global timestamp is 0.

Given a channel state $r : C \to 2^{\mathbb{R}_{\geq 0}}$, let $\tau_{\min}(r) = \min \bigcup_{c \in C} r(c)$. That is, $\tau_{\min}(r)$ is the minimum timestamp among all currently pending events in $r$. Given actor $a \in A$ and $r$, we denote by $\tau_{\min}(a, r)$ the minimum timestamp among all currently pending events in the input channel(s) of $a$ at $r$. That is, $\tau_{\min}(a, r) = \min \bigcup_{c \in C^{in}(a)} r(c)$. Note that $\tau_{\min}(a, r) \geq \tau_{\min}(r)$ for any $a, r$. By convention we set $\min \emptyset = \infty$. This implies that for an empty channel state $r$, we have $\tau_{\min}(r) = \infty$. Also, if $C^{in}(a) = \emptyset$, that is, if $a$ has no input channels, then $\tau_{\min}(a, r) = \infty$ for all $r$.

We say that an actor $a \in A$ is *enabled* at state $s = (r, t)$, denoted *enabled*$(a, s)$, if $\tau_{\min}(a, r) = \tau_{\min}(r) = t$. That is, $a$ is enabled at $s$ if there is at least one event

pending in one of the inputs of $a$ which has timestamp $\tau$ no greater than the smallest timestamp in $r$ and $\tau$ agrees with the global time $t$. We say that $a$ is *strongly enabled* at $s$ if $enabled(a, s)$ and there is no actor $b \neq a$ such that $enabled(b, s)$ and $b \prec a$. That is, $a$ is strongly enabled at $s$ if it is enabled and there is no actor $b$ which is also enabled at $s$ and which comes before $a$ according to $\prec$.

We next define the operation of *firing* an actor, and the effect that this has on the state. Intuitively, firing an actor $a$ at state $s = (r, t)$ consists in removing the event $\tau_{\min}(a, r)$ from all input channels of $a$ that contain this event, and adding the event $\tau_{\min}(a, r) + D(a)$ to each output channel of $a$. Formally, we define an auxiliary function $f(a, r, d)$ which, given actor $a \in A$, channel state $r$, and delay $d \in \mathbb{R}_{\geq 0}$, returns a channel state $r'$ defined as follows:

$$r'(c) = \begin{cases} r(c) - \{\tau_{\min}(a, r)\} & \text{if } c \in C^{in}(a) \\ r(c) \cup \{\tau_{\min}(a, r) + d\} & \text{if } c \in C^{out}(a) \\ r(c) & \text{otherwise.} \end{cases} \quad (1)$$

We are now ready to define the transition relation $\rightarrow$ of $TTS(G, r_0)$. $\rightarrow$ has two types of transitions: *discrete transitions* of the form $s \xrightarrow{a} s'$, such that $a$ is strongly-enabled in state $s = (r, t)$ and $s' = (r', t)$ with $r' = f(a, r, D(a))$, and *timed transitions* of the form $s \xrightarrow{\delta} s'$, where $s, s' \in S$, $a \in A$, and $\delta \in \mathbb{R}_{\geq 0}$. Timed transitions are enabled at a state $s = (r, t)$ when no discrete transition is enabled at $s$ and when $r$ is not empty. In that case, it must be $t < \tau_{\min}(r)$ and $\tau_{\min}(r) \neq \infty$. Then, a timed transition $s \xrightarrow{\delta} (r, t')$ occurs, with $\delta = \tau_{\min}(r) - t$ and $t' = t + \delta = \tau_{\min}(r)$.

**Remarks:** (1) Global time is not affected by a discrete transition and channel state is not affected by a timed transition. (2) In $TTS(G, r_0)$ there cannot be two timed transitions in a row. (3) According to the rule $\tau_{\min}(a, r) = \infty$, source actors are never enabled, and therefore never fire. We use source actors simply to allow for initial events at the inputs of some actors.

A state $s = (r, t)$ is a *deadlock*, that is, has no outgoing transitions, iff $r$ is empty, that is, for every $c \in C$, $r(c) = \emptyset$.

An *execution* of $TTS(G, r_0)$ is a sequence of states $\rho = s_0, s_1, ...$ such that there is a (discrete or timed) transition from every $s_i$ to $s_{i+1}$. We require $\rho$ to be *maximal*, that is, either infinite or ending in a deadlock state. Note that if the DDE model contains a loop that is reachable from some initial event, there will not be a deadlock state.

**Lemma 1.** *Let $\rho = s_0, s_1, ...$ be an execution of $TTS(G, r_0)$ and let $a$ be an actor of $G$. For any transitions $s_i \xrightarrow{a} s_{i+1}$ and $s_j \xrightarrow{a} s_{j+1}$ in $\rho$ such that $s_i = (r_i, t_i)$, $s_j = (r_j, t_j)$, and $i < j$, we have $t_i < t_j$.*

Lemma 1 states that every actor is fired in timestamp order, and in particular, that it cannot be fired more than once before time elapses.

$TTS(G, r_0)$ has several deterministic properties. First, by definition, if $s \xrightarrow{a} s'$ then there is no $s'' \neq s'$ such that $s \xrightarrow{a} s''$. Second, $TTS(G, r_0)$ has the so-called "diamond property":

**Lemma 2.** *If $s \xrightarrow{a} s_1$ and $s \xrightarrow{b} s_2$, then there is a unique $s'$ such that $s_1 \xrightarrow{b} s'$ and $s_2 \xrightarrow{a} s'$.*

Let $\rho = s_0, s_1, \ldots$ be an execution of $TTS(G, r_0)$ where $G = (A, C, D)$ and $s_i = (r_i, t_i)$. The *signal* of a channel $c \in C$ under execution $\rho$, denoted $\sigma_c^\rho$, is defined to be the set of all events occurring in $c$ along the entire execution: $\sigma_c^\rho = \bigcup_{i \in \mathbb{N}} r_i(c)$.

**Lemma 3 (Kahn property [12]).** *For any $c \in C$ and any two executions $\rho_1$ and $\rho_2$, $\sigma_c^{\rho_1} = \sigma_c^{\rho_2}$.*

Because of the Kahn property, we can write $\sigma_c$ for the unique signal of a channel $c$. This can be viewed as the denotational semantics of DDE models.

## 3 Boundedness of DDE

In this section we study boundedness of the state-space of DDE models. Let us begin with an illustrative example. Figure 2 shows a DDE model $G = (A, C, D)$, with $A = \{a_1, a_2\}$, $C = \{c_1 : (a_1, a_1), c_2 : (a_1, a_2)\}$, $D(a_1) = P$, and $D(a_2) = 0$. This model captures a periodic source with period $P$, generating events at times $P, 2P, \cdots$. The model includes actor $a_1$ which delays its input by $P$ and a sink actor $a_2$. If the delay of an actor is non-zero, it is drawn inside the actor. Zero delays are not drawn. The model has two channels, $c_1, c_2$. Channel $c_1$, a self-loop of $a_1$, contains an initial event with timestamp 0. This is the only initial event in the system (empty initial event sets on channels are not drawn). The initial event models the seed of the periodic source. Actor $a_1$ adds a delay of $P$ to the event's timestamp, outputs the event to $c_2$, which represents the source's output, and starts anew a cycle where the initial event is replaced with an event with timestamp $P$.

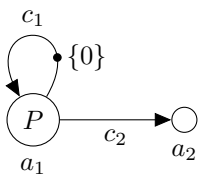



**Fig. 2.** Periodic clock.

The initial channel state is $r_0 = \{(c_1, \{0\}), (c_2, \{\})\}$. A prefix of a path in the transition system $TTS(G, r_0)$ is the following:

$$
\begin{aligned}
&s_0 : (c_1 : \{0\}, c_2 : \{\}, t = 0) && \xrightarrow{a_1} s_1 : (c_1 : \{P\}, c_2 : \{P\}, t = 0) && \xrightarrow{P} \\
&s_2 : (c_1 : \{P\}, c_2 : \{P\}, t = P) && \xrightarrow{a_2} s_3 : (c_1 : \{P\}, c_2 : \{\}, t = P) && \xrightarrow{a_1} \\
&s_4 : (c_1 : \{2P\}, c_2 : \{2P\}, t = P) && \xrightarrow{P} s_5 : (c_1 : \{2P\}, c_2 : \{2P\}, t = 2P) && \xrightarrow{a_1} \\
&s_6 : (c_1 : \{3P\}, c_2 : \{2P, 3P\}, t = 2P) && \xrightarrow{a_2} s_7 : (c_1 : \{3P\}, c_2, \{3P\}, t = 2P) && \xrightarrow{P} \cdots
\end{aligned}
$$

Note that in state $s_2$ there are two events with timestamps equal to $\tau_{\min} = P$ but they are both strongly enabled since it is neither the case that $a_1 \prec a_2$ nor $a_2 \prec a_1$. Furthermore, it is easy to see that the signal in $c_1$ is $\sigma_{c_1} = \{i \cdot P \mid i \in \mathbb{N}\}$ and the signal in $c_2$ is $\sigma_{c_2} = \{i \cdot P \mid i \in \mathbb{N}_{>0}\}$.

As it can be seen from the above example, $TTS(G, r_0)$ is generally infinite-state. There are two potential sources of infinity of state-space in DE models.

First, the timestamps may grow unbounded, as is the case with the above example. Second, it is unclear whether the set of events on each channel remain bounded. This is true in the above example, but is it generally true? In Section 3.1 we show that this is true for all DDE models. Then in Section 3.2 we show how timestamps can also be bounded.

### 3.1 Bounding the number of events in the channels

Let us begin by providing some intuition about why the number of events in an execution of $TTS(G, r_0)$ remains bounded.

Consider the example in Figure 3. The set of events produced by "loop1" in channel $c_1$ is $\{i \cdot P \mid i \in \mathbb{N}\}$. Each new event with timestamp $t$ that enters "loop2" from "loop1", will result in an infinite set of events $\{t + j \cdot D \mid j \in \mathbb{N}_{>0}\}$ in channel $c_2$. Therefore the set of all events in channel $c_2$ will be $\{i \cdot P + j \cdot D \mid i, j \in \mathbb{N}_{>0}\}$.

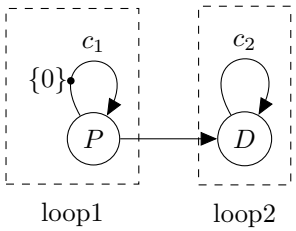



**Fig. 3.** Loop example.

Because $P, D \in \mathbb{N}$, the timestamp of any event that appears in $c_2$ can be written as $k \cdot \mathsf{gcd}(P, D)$ for some $k$. In fact, there exists $n$, such that for all $k > n$, there exist positive $i$ and $j$ such that $k \cdot \mathsf{gcd}(P, D) = i \cdot P + j \cdot D$. So eventually all multiples of $\mathsf{gcd}(P, D)$ appear as timestamps of events in $c_2$.

Note that in every reachable state $s = (r, t)$ of $TTS(G, r_0)$, for $G = (A, C, D)$, an upper bound on the timestamp of any event is $\tau_{\min}(r) + \max\{D(a) \mid a \in A\}$, and a lower bound is $\tau_{\min}(r)$. Hence, because event timestamps in $c_2$ are separated by at least $\mathsf{gcd}(D, P)$, the number of events in $c_2$ satisfies

$$|r(c_2)| \leq \left\lceil \frac{\max\{D(a) \mid a \in A\}}{\mathsf{gcd}(D, P)} \right\rceil \text{ in any state } s = (r, t).$$

In general, let $G = (A, C, D)$ be a DE model and let $r_0$ be an initial channel state for $G$. Let $TTS(G, r_0) = (S, s_0, \rightarrow)$. Consider a state $s = (r, t) \in S$. Recall that $r$ is a function $r : C \rightarrow 2^{\mathbb{R}_{\geq 0}}$. The *size* of $r$, denoted $|r|$, is defined to be

$$|r| := \sum_{c \in C} |r(c)|$$

**Theorem 1 (Boundedness of channels).** *There exists $K \in \mathbb{N}$ such that for every reachable state $s = (r, t)$ of $TTS(G, r_0)$, $|r| \leq K$.*

### 3.2 Bounding timestamps

In $TTS(G, r_0)$ timestamps of events can still grow unbounded. Moreover, there is the additional global timestamp which grows unbounded too. Nevertheless, it is easy to see how to transform $TTS(G, r_0)$ in order to obtain an equivalent

*bounded timed transition system*, which we will denote $BTS(G, r_0)$. To define $BTS(G, r_0)$, we introduce some notation. Let $s = (r, t)$ be a state of $TTS(G, r_0)$. Let $\delta \in \mathbb{R}_{\geq 0}$ be such that $\delta \leq \tau_{\min}(r)$. Then we denote by $r - \delta$ the new channel state $r'$ obtained from $r$ by decrementing all timestamps in $r$ by $\delta$.

We are now ready to define $BTS(G, s_0)$. Its states are channel states, that is, the global timestamp is dropped. On the other hand, $BTS(G, r_0)$ has both discrete and timed transitions, like $TTS(G, r_0)$. A timed transition in $BTS(G, r_0)$ has the form $r \xrightarrow{\delta}_b r'$ where $\delta = \tau_{\min}(r)$ and $r' = r - \tau_{\min}(r)$. A discrete transition in $BTS(G, s_0)$ has the form $r \xrightarrow{a}_b r'$ with $r' = f(a, r, D(a))$, such that $\tau_{\min}(a, r) = \tau_{\min}(r) = 0$. That is, in $BTS(G, r_0)$, we keep track of time elapsing by appropriately decrementing the timestamps of pending events.

**Theorem 2.** *The set of reachable states of $BTS(G, r_0)$ is finite.*

Is it easy to show that a bisimulation exists between TTS and BTS. In particular, let $s = (r, t)$ be a reachable state of $TTS(G, r_0)$. It can be easily shown, by induction on the transition relation of $TTS(G, r_0)$, that $s$ satisfies $t \leq \tau_{\min}(r)$. We define the relation $R$ between states of $TTS(G, r_0)$ and states of $BTS(G, r_0)$, so that $R$ contains all pairs $((r, t), r - t)$. It can be checked that $R$ is a bisimulation relation.

## 4 Extended discrete-event models

In this section we introduce extensions to the DDE model.

### 4.1 Non-deterministic DE

The *non-deterministic DE model* (NDE) extends DDE by allowing actors with variable delays, specifically intervals.

The syntax of an NDE model is almost the same as that of a DDE model. It is a labeled graph $G = (A, C, D)$, with $A$ and $C$ being as in a DDE model, and $D$ associating an interval instead of a fixed value to each actor. Intervals must be non-empty, and can be of the form $[l, u]$, $(l, u)$, $(l, \infty)$, and so on, for $l, u \in \mathbb{N}$. When the interval is $[l, l]$ we simply write $D(a) = l$. We allow loops, but require that every loop visits at least one actor $a$ such that $l(a) \geq 1$, where $l(a)$ is the lower bound of $D(a)$. The partial order $\prec$ is also defined in NDE, so that $a \prec a'$ iff there exists a path from $a$ to $a'$ such that for any actor $a''$ in the path (including $a$ but excluding $a'$) we have $l(a'') = 0$.

The semantics of NDE is defined as a timed transition system, as with DDE. Given an NDE graph $G$, and an initial channel state $r_0$, $TTS(G, r_0)$ is defined to be the tuple $(S, s_0, \rightarrow)$ where $S$ and $s_0$ are as in DDE, and the transition relation $\rightarrow$ contains both discrete and timed transitions. A discrete transition of $TTS(G, r_0)$ is of the form $(r, t) \xrightarrow{a} (r', t)$ where $a$ is strongly enabled in $(r, t)$ and $r' = f(a, r, d)$, for some $d \in D(a)$. The definition of strongly enabled for NDE is the same as in DDE and uses the partial order $\prec$ as defined above. The timed transitions of $TTS(G, r_0)$ are defined in the same way as in DDE.

We point out that the above semantics allows to "reorder" events, in the sense that an event produced in a channel could have timestamp smaller than the events already in the channel. However, execution of actors is still guaranteed to happen in timestamp order. Also, since we are not currently using multisets, if an event is added to a channel which already has an event with the same timestamp then the two events are merged into one.

### 4.2 DE with timed automata

The *DE with timed automata model* (DETA) extends DDE by allowing actors to be modeled as timed automata. This extension allows actors in a discrete-event program to model environment behavior as well as have more elaborate internal behavior than DDE and NDE.

Like DDE and NDE, a DETA model is represented by a labeled graph. In the case of DETA, a label is either a fixed delay or a timed automaton (TA). Formally, a DETA model is a graph $G = (A, C, L)$, with $A$ and $C$ being as in a DDE model, and $L$ being a labeling function which maps every actor $a \in A$ to either a delay $d \in \mathbb{N}$, or a TA $M = (Q, q_0, X, I, E)$, where:

- $Q$ is the set of *locations* of $M$, and $q_0 \in Q$ is its *initial location.*
- $X$ is the set of *clocks* of $M$. Both $Q$ and $X$ are finite sets.
- $I$ is the *invariant function* which maps every $q \in Q$ to a simple convex constraint of the form $\bigwedge_i x_i \leq k_i$, where $x_i \in X$ are clocks and $k_i \in \mathbb{N}$ are constants.
- $E$ is the set of *transitions* of $M$. A transition is a tuple $h = (q, c, q', \phi, X')$ where:
  - $q, q' \in Q$: $q$ is the source and $q'$ the destination location of $h$.
  - $c \in C(a)$, i.e., $c$ is either an input or an output channel of actor $a$.
  - $\phi$ is a simple constraint on clocks, called the *guard* of $h$.
  - $X' \subseteq X$ is a subset of clocks to be *reset* by $h$.

We require that every TA $M$ in a DETA model be *receptive*, that is, able to accept any input event at any state. Formally, for every location $q$ of $M$, and for every input channel $c$ of $a$, the union of all guards of all outgoing transitions from $q$ labeled with $c$ must cover the whole space of clock valuations, that is, must be equivalent to the guard *true*.

We allow loops in DETA models, but we assume conservatively that the delay introduced by TA actors could be zero. Therefore we require that every loop visits at least one constant delay actor with delay $\geq 1$. The partial order $\prec$ is defined for DETA in the same way as for DDE, by treating TA actors like zero-delay actors.

Before defining the semantics of DETA models we briefly recall the semantics of TA. A state of a TA $M$ is a pair $(q, v)$ where $q \in Q$ is a location and $v$ is a *clock valuation*, that is, a function $v : X \to \mathbb{R}_{\geq 0}$ mapping every clock of $M$ to a non-negative real value. We will use the term *TA state* for a pair $(q, v)$, to avoid confusion with states of DE models, which we sometimes for clarity call *channel*

*states*. The initial TA state of $M$ is defined to be $(q_0, \mathbf{0})$, where $\mathbf{0}$ is the valuation assigning 0 to all clocks. $M$ defines two types of transitions on this state-space: discrete and timed transitions. A discrete transition is possible from TA state $(q, v)$ to TA state $(q', v')$, denoted $(q, v) \xrightarrow{c}_M (q', v')$, if $M$ has a transition $h = (q, c, q', \phi, X')$ such that: (1) $v$ satisfies the guard $\phi$, denoted $v \models \phi$; (2) $v' = v[X' := 0]$, which means that $v'(x) = 0$ if $x \in X'$ and $v'(x) = v(x)$ otherwise; and (3) $v'$ satisfies the invariant of the destination location $q'$, denoted $v' \models I(q')$. A timed transition of delay $\delta \in \mathbb{R}_{\geq 0}$ is possible from TA state $(q, v)$ to TA state $(q, v')$, denoted $(q, v) \xrightarrow{\delta}_M (q, v')$, if: (1) $v' = v + \delta$, which means that $v'(x) = v(x) + \delta$ for all $x \in X$; and (2) $v' \models I(q)$. The latter condition, together with our assumption on the form of invariants, ensures that the progress of time from $v$ to $v'$ does not violate any urgency constraints at location $q$. Note that, since $I(q)$ is downwards-closed, $v + \delta \models I(q)$ implies that for any $\delta' \leq \delta$, we also have $v + \delta' \models I(q)$.

We are now ready to define the semantics of DETA models. Consider a DETA model $G = (A, C, L)$ and an initial channel state $r_0$. Let $A_{TA}$ be the subset of $A$ such that $a \in A_{TA}$ iff $L(a)$ is a TA. For $a \in A_{TA}$, we denote the TA $L(a)$ by $M_a$. Then, $G$ and $r_0$ define the timed transition system $TTS(G, r_0)$. A state of $TTS(G, r_0)$ is a triple $(r, w, t)$ where $r$ is a channel state, $w$ is a total function mapping actors in $A_{TA}$ to TA states, and $t \in \mathbb{R}_{\geq 0}$ is a global timestamp. For given $a \in A_{TA}$, $w(a)$ represents the TA state which $M_a$ is currently at.

Like the other timed transition systems defined earlier, $TTS(G, r_0)$ has two types of transitions: discrete and timed. A discrete transition has the form $(r, w, t) \xrightarrow{a} (r', w', t)$, for $a \in A$, and is possible if:

- either $a \notin A_{TA}$, that is, $L(a) \in \mathbb{N}$, in which case $r' = f(a, r, L(a))$ and $w' = w$;
- or $a \in A_{TA}$, in which case
  1. either $a$ has an input channel $c$ such that $t \in r(c)$, in which case:
     (a) $r'$ is obtained from $r$ by removing the event with timestamp $t$ from $c$, that is, $r'(c) = r(c) - \{t\}$ and $r'(c') = r(c')$ for all $c' \neq c$.
     (b) $w'$ is obtained from $w$ by having $M_a$ take the discrete transition $w(a) \xrightarrow{c}_{M_a} w'(a)$ in reaction to the event in $c$, and having all other TA retain their state, that is, $w'(a') = w(a)$ for all $a' \in A_{TA}, a' \neq a$.
  2. or $a$ has an output channel $c$ such that $w(a) \xrightarrow{c}_{M_a} w'(a)$ and for all $a' \in A_{TA}$ s.t. $a' \neq a$, we have $w'(a') = w(a)$. In this case, $r'(c) = r(c) \cup \{t\}$ and $r'(c') = r(c')$ for all $c' \neq c$.

The case $a \notin A_{TA}$ corresponds to the case where a standard DDE actor fires, that is, an actor introducing a deterministic delay. The case $a \in A_{TA}$ corresponds to the case where a TA actor fires, that is, makes a discrete transition. In this case, the following subcases are possible:

- Either $a$ consumes an event from an input channel and reacts to it (Case 1). Note that since $M_a$ is assumed to be receptive, the transition $w(a) \xrightarrow{c}_{M_a} w'(a)$ is guaranteed to exist. Also note that it is by definition impossible for

a TA actor to consume multiple events from multiple input channels in a single transition. This is true even when all these events may have the same timestamp. On the other hand, in that case the TA actor will consume all these events in a series of discrete *simultaneous* transitions, that is, without time passing in-between these transitions.

– Or $a$ "spontaneously" produces an event to an output channel (Case 2).

A timed transition in $TTS(G, r_0)$ has the form $(r, w, t) \xrightarrow{\delta} (r, w', t + \delta)$, for $\delta \in \mathbb{R}_{\geq 0}$, and is possible if:

1. $t + \delta \leq \tau_{\min}(r)$; and
2. for all $a \in A_{TA}$, $M_a$ has a timed transition by $\delta$, that is, $w(a) \xrightarrow{\delta}_{M_a} w'(a)$ is a valid transition.

That is, a timed transition by $\delta$ is possible if it is possible for every TA in the system to let time elapse by $\delta$, and also if this does not violate the urgency of any pending event in the system. Note that it is possible in DETA to have several timed transitions in a row.

An example of a DETA model is provided in the left part of Figure 4. There are four actors in this model, one of which, $a_2$, is a TA actor. The automaton for $a_2$ has two locations, $q_0, q_1$, and a single clock $x$. The invariant at $q_1$ is $x \leq 1$, whereas the invariant at $q_0$ is *true* and therefore not shown. The guard in the transition from $q_0$ to $q_1$ is also *true*, and not shown either. The label $x := 0$ means that $x$ is reset on the corresponding transition (absence of such a label means that the clock is not reset). In the transitions of the automaton, we use the label $c?$ instead of $c$ when $c$ is an input channel, to emphasize the fact that the actor consumes an event from $c$. Similarly, we use $c!$ when $c$ is an output channel, to emphasize the fact that the actor produces an event in $c$. A sample execution of this DETA model is provided in the right part of Figure 4.

*Executions and signals in NDE and DETA:* The notions of executions and signals can be easily extended from DDE to NDE and DETA models. Because of non-determinism in both NDE and DETA, Lemmas 2 and 3 do not hold in neither

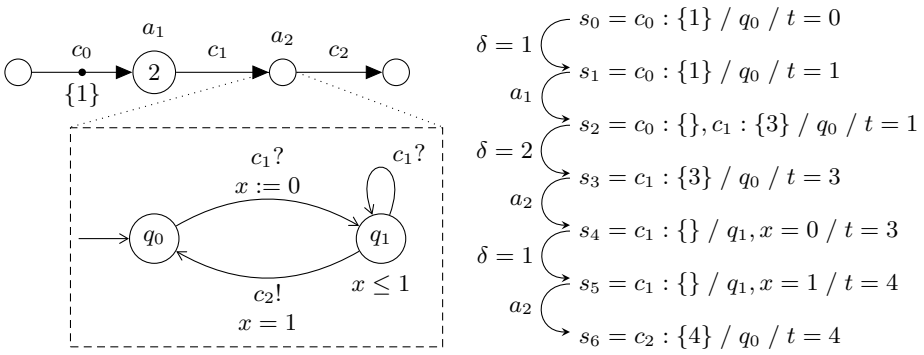



**Fig. 4.** A DETA model (left) and a sample execution (right).

NDE nor DETA. This means in particular that the signal $\sigma_c^\rho$ of a given channel $c$ in these models generally depends on the execution $\rho$. For NDE and DETA models, we define $\sigma_c$ to be the union of $\sigma_c^\rho$ over all executions $\rho$.

*Unboundedness of NDE and DETA:* Boundedness does not hold for neither NDE nor DETA models. We can show that if we feed a variable delay with a periodic stream of events we can construct a sporadic stream which, in turn, if fed into a periodic loop causes Theorem 1 to fail. Figure 5 illustrates the idea. In this model, the variable delay is implemented as a TA, resulting in a DETA model. The variable delay can also be implemented as an NDE actor $a$ with $D(a) = [l, u]$, resulting in an NDE model.

More precisely, assume that the input stream of $a$ has period $P$, as shown in Figure 5, and that $P > u$. Then the TA shown in the figure correctly implements a variable delay and every event coming out of the loop with period $P$ will be given a variable delay $[l, u]$. Let $a$ add delay $l + \frac{u-l}{i}$ to its $i^{\text{th}}$ input event. This will result in an output stream of events $\{l+u-l, P+l+\frac{u-l}{2}, 2\cdot P+l+\frac{u-l}{4}, \ldots\}$.

In general, if a signal of the form $\{i \cdot P + \frac{x}{2^i} \mid i \in \mathbb{N}\}$ is fed into a loop with delay $D$, then the signal in the loop will contain events $\{i \cdot D + j \cdot P + \frac{x}{2^j} \mid i, j \in \mathbb{N}\}$. This set of events has the property that there is no bound $K \in \mathbb{N}$ such that for every $n$ the number of events in window $[n, n+1]$ is less than $K$. Intuitively the reason is that for any $K$, an $n$ can be found such that the equation $i \cdot D + j \cdot P = n$ has more than $K$ solutions, and since, for large enough $j$, $x/2^j < 1$, the window $[n, n+1]$ will contain more than $K$ events.

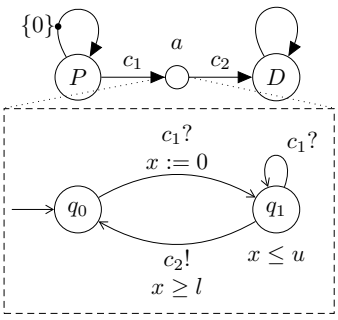



**Fig. 5.** Unbounded DETA model.

## 5 Verification

We begin by defining the types of verification queries that we are interested in.

Let $G$ be a DE model (i.e., a DDE, NDE, or DETA model) with set of channels $C$ and let $r_0$ be an initial channel state. Let $\rho$ be an execution of $TTS(G, r_0)$. Recall that $\sigma_c^\rho$, for channel $c \in C$, denotes the set of all events (timestamps) that occur in $c$ along execution $\rho$ in $TTS(G, r_0)$. A *signal query* is a query of the form "does $\sigma_c^\rho$ satisfy some property $\phi$?", where $\phi$ is a property written in (some subclass of) first-order logic. For instance, the property "an event occurs in $c$" can be written as $\phi := \exists \tau : \tau \geq 0$. The property "two events occur in $c$ at two distinct times in the interval $[1, 2]$" can be written as $\exists \tau_1, \tau_2 : \tau_1 \neq \tau_2 \wedge 1 \leq \tau_1, \tau_2, \leq 2$. The property "two events occur in $c$ separated by at most 1 time unit" can be written as $\exists \tau_1, \tau_2 : |\tau_1 - \tau_2| \leq 1$.

We are also interested in queries which involve states of $TTS(G, r_0)$. A *state query* asks whether there exists a reachable state $s = (r, t)$ such that $r$ satisfies some property $\psi$. Again, we can imagine various types of properties $\psi$. For

example, given constant $k \in \mathbb{N}$, $\psi$ could be the expression $|r| > k$, which states that there are more than $k$ events pending in the system, or the expression $|r(c)| > k$, for given $c \in C$, which states that there are more than $k$ events pending on channel $c$. $\psi$ could also be an expression such as those mentioned for signal queries above, stating, for example, that $r$ contains an event with a certain timestamp or timestamp bounds, two events with a certain time difference, etc.

Channel signals are denotational semantics of DE models and signal queries allow to express natural properties on these. State queries are also important, as they refer to system snapshots as well as to implementation properties such as buffer space requirements. In the rest of this section we discuss how signal and state queries can be automatically checked on DDE models.

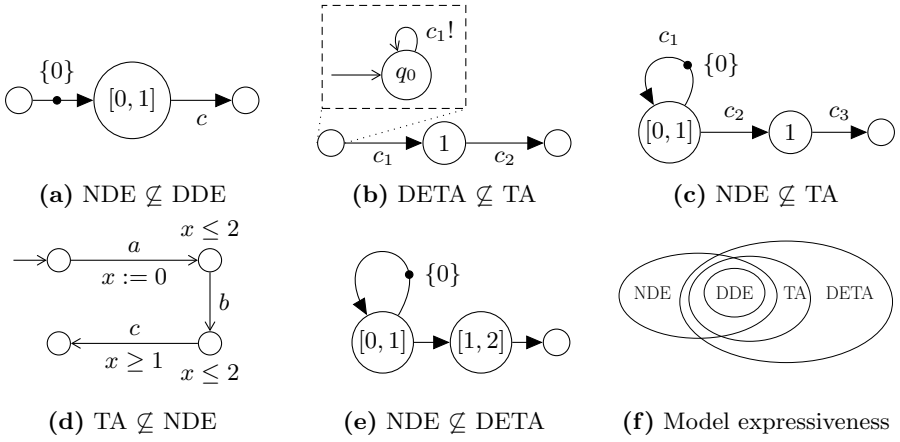
First, consider signal queries. They can be checked with the help of a *lasso* derived from BTS. This lasso is a finite and deterministic transition system (deterministic in the sense that every state has at most one successor), derived from BTS by merging all enabled discrete transitions from a given state $s$ into a single *supertransition* where all corresponding actors fire. The diamond property (which by the bisimulation property also holds on BTS) ensures that this transformation is valid. We can analyze this lasso and compute, for every channel $c$, an affine expression that describes $\sigma_c$. Then we can reduce the problem of checking whether $\sigma_c$ satisfies a signal query $\phi$ to an SMT (satisfiability modulo theory) problem. For example, the affine expression for channel $c_2$ for the example of Figure 3 is $i \cdot P + j \cdot D$, with $i, j \in \mathbb{N}_{>0}$. Checking whether there exist two events $\tau_1, \tau_2$ in $\sigma_{c_2}$ such that $\tau_1 - \tau_2 = 5$, can then be reduced to checking satisfiability of the expression $\tau_1 = i_1 \cdot P + j_1 \cdot D \wedge \tau_2 = i_2 \cdot P + j_2 \cdot D \wedge \tau_1 - \tau_2 = 5$.

Second, for state queries, we can again use the lasso and the bisimulation of BTS and TTS to compute an affine expression characterizing the set of timestamps on a *per state* basis. We can then reduce the problem of whether there exists a reachable state satisfying a property $\psi$ to a series of SMT problems, one for every affine expression computed for every state in the lasso.

## 6 Expressiveness

In this section we discuss how the various DE models introduced above are related to each other, as well as to timed automata, in terms of expressiveness. We write $\mathcal{A} \subseteq \mathcal{B}$ if for every model $G$ in formalism $\mathcal{A}$ there exists a model $G'$ in formalism $\mathcal{B}$ such that $G$ and $G'$ are equivalent in terms of denotational semantics, i.e., channel signals. More precisely, $G$ and $G'$ are equivalent if they refer to the same set of channels $C$, and for every $c \in C$, $\sigma_c^G = \sigma_c^{G'}$, where $\sigma_c^G, \sigma_c^{G'}$ are the signals of $c$ in $G, G'$, respectively.

To be able to compare the DE models with timed automata, we view TA as a subclass of DETA models. Concretely, suppose $M$ is a TA whose transitions are labeled with $c_1, c_2, ..., c_n$. Then $M$ can be seen as a DETA model with $n+1$ actors, $a, a_1, ..., a_n$, where $a$ is a source actor labeled by $M$, and $a_1, ..., a_n$ are sink actors connected to $a$. In this interpretation, every label $c_i$ of $M$ is seen as a distinct output channel of $a$.




**(a)** NDE ⋢ DDE **(b)** DETA ⋢ TA **(c)** NDE ⋢ TA

**(d)** TA ⋢ NDE **(e)** NDE ⋢ DETA **(f)** Model expressiveness

**Fig. 6.** Models used in expressiveness discussion.

The expressiveness results, summarized in Figure 6(f), are as follows:

**DDE ⊊ NDE:** DDE ⊆ NDE because the fixed delay $d$ can be expressed as the interval $[d, d]$. NDE ⋢ DDE because NDE allows non-deterministic behavior but DDE does not. Indeed, the NDE model of Figure 6(a) produces a single event on channel $c$ at time $t \in [0, 1]$, but this is impossible to express in DDE.

**DDE ⊊ TA:** TA ⋢ DDE because TA allows non-deterministic behavior but DDE does not. The example of Figure 6(a) can be easily constructed with TA. To see why DDE ⊆ TA, consider the lasso defining the signals of a DDE model, discussed in Section 5. The affine expressions describing the channel signals can be directly transformed into parallel compositions of simple TA with periodic self-loops. For example, $(2 + 3 \cdot i) \cup (3 + 7 \cdot j)$ can be trivially transformed into the parallel composition of two TA.

**DDE ⊊ DETA:** DDE ⊆ DETA because every DDE model is by definition a DETA model (one that has no TA actors). DETA ⋢ DDE again because of non-determinism.

**TA ⊊ DETA:** As defined above, TA is by definition a subclass of DETA. To see that DETA ⋢ TA, consider the DETA model shown in Figure 6(b). In this model, every event produced by the TA on channel $c_1$ is delayed by the constant delay actor by exactly 1 time unit. Since there is no bound on the number of events that can be produced on $c_1$ in a time interval of size 1, an equivalent TA model would require an unbounded number of clocks.

**NDE ⋢ TA:** To see this, consider the example of Figure 6(c). Similarly to the model of Figure 6(b), in this model the number of events that can be produced in an interval of size 1 on channel $c_2$ is unbounded, and for every such event a TA implementation would require a separate clock to produce the corresponding event on channel $c_3$.

We also conjecture that the TA of Figure 6(d) cannot be implemented in NDE. This TA produces three events, $a, b, c$, in that order, with the constraint that the distance between $a$ and $c$ is in the interval $[1, 2]$. This behavior requires

both non-deterministic delays and some form of synchronization to ensure that $b$ occurs before $c$, which does not appear to be implementable in NDE.

We finally conjecture that the NDE model of Figure 6(e) cannot be implemented in DETA. The loop in the model can produce unbounded bursts of events in a finite window. A variable delay needs to be introduced for every such event. In DETA, variable delays can only be implemented using timed automata, which only have a finite number of clocks.

## 7 Related work

The term *discrete-event systems* (DES) is often used to denote untimed models based on automata, Petri nets, and related formalisms, with a focus on controller synthesis and similar problems [17,4]. In this paper we study timed DE models.

Timed and actor-oriented DE models have been considered previously in a number of works, but with a different focus than that of our paper. [20,14,16,5,15] focus on the semantics of timed systems. [10] focuses on compositionality and preservation of properties such as worst-case throughput or latency. [3] presents a translation of Ptolemy DE models to Real-Time Maude for the purposes of verification, but does not study the verification problem *per se*. Rebeca is an actor-oriented language where actors can specify timed behavior using statements such as *delay* [18]. However, decidability of model-checking for timed Rebeca has not been investigated. [6] extends finite-state automata with delay blocks and examines the expressiveness of the resulting timed languages.

A large body of research exists on the real time calculus (RTC) [19]. RTC focuses on performance properties for which interesting analytic and modular techniques can be derived. On the other hand, RTC models have relatively limited expressiveness compared to state-based models such as TA and the analysis results are generally approximations. Recent works on RTC include techniques for models combining analytic components as used in standard RTC with TA [13] and techniques to handle networks with cyclic dependencies [11].

DE models have a significant difference from time(d) Petri nets, even though at first sight they may appear similar. In Petri nets, a transition fires only if enough tokens are available in *every* input place, whereas in DE, an actor fires in timestamp order, and may consume tokens only from *some* input channels.

## 8 Perspectives

The verification problems for NDE and DETA remain open. We are currently exploring ideas to constrain the model to regain, or statically check for, boundedness, which would enable transformation of bounded DETA models to timed automata. We are also currently working on extracting affine expressions directly from DDE models (without the use of lassos) and then extending this technique to NDE, which would allow verification of signal queries on NDE despite unboundedness. TA are another possible symbolic representation of signals, natural in DETA models. It is easy to see how to transform TA signal representations by fork, join, constant- and variable-delay actors, but not how to compute fixpoints which seems needed for general cyclic networks.

Another direction for future work is investigating model-checking of general temporal logics against DE models, or coming up with new logics especially designed for DE models.

Another direction is to enrich expressiveness of DDE and NDE models, for instance, by adding control-expressive actors such as synchronizers, which from the comparison between NDE and TA appear important. Adding values to events is another possibility for extending DE models in general, including DETA models.

## References


1. G. Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems*. The MIT Press Series in Artificial Intelligence. MIT Press, Cambridge, MA, 1986.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. K. Bae, P. Csaba Olveczky, T. H. Feng, E. A. Lee, and S. Tripakis. Verifying Hierarchical Ptolemy II Discrete-Event Models using Real-Time Maude. *Science of Computer Programming*, 77(12):1235 – 1271, 2012.
4. C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
5. A. Cataldo, E. Lee, X. Liu, E. Matsikoudis, and H. Zheng. A constructive fixed-point theorem and the feedback semantics of timed sy stems. In *WODES*, 2006.
6. Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Finite automata with time-delay blocks. In *EMSOFT '12*, pages 43–52. ACM, 2012.
7. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*. Springer, 1989.
8. J.C. Eidson, E.A. Lee, S. Matic, S.A. Seshia, and J. Zou. Distributed real-time software for cyber-physical systems. *Proceedings of the IEEE*, 100(1):45–59, 2012.
9. J. Eker, J. Janneck, E. A. Lee, et al. Taming heterogeneity – the Ptolemy approach. *Proc. IEEE*, 91(1), 2003.
10. M. Geilen, S. Tripakis, and M. Wiggers. The Earlier the Better: A Theory of Timed Actor Interfaces. In *HSCC*. ACM, 2011.
11. B. Jonsson, S. Perathoner, L. Thiele, and W. Yi. Cyclic dependencies in modular performance analysis. In *EMSOFT '08*, pages 179–188. ACM, 2008.
12. Gilles Kahn. The semantics of a simple language for parallel programming. 1974.
13. K. Lampka, S. Perathoner, and L. Thiele. Analytic real-time analysis and timed automata: a hybrid method for analyzing embedded real-time systems. In *EMSOFT*, pages 107–116. ACM, 2009.
14. E.A. Lee. Modeling concurrent real-time processes using discrete events. *Annals of Software Engineering*, 7(1):25–45, 1999.
15. X. Liu and E. A. Lee. CPO semantics of timed interactive actor networks. *Theoretical Computer Science*, 409(1):110–125, 2008.
16. X. Liu, E. Matsikoudis, and E. A. Lee. Modeling timed concurrent systems. In *CONCUR*, volume LNCS 4137. Springer, 2006.
17. P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, January 1989.
18. M. Sirjani and M. M. Jaghoori. Ten Years of Analyzing Actors: Rebeca Experience. In *Formal Modeling: Actors, Open Systems, Biological Systems*, pages 20–56, 2011.
19. L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *ISCAS*, pages 101–104, 2000.
20. R. K. Yates. Networks of real-time processes. In *CONCUR*. Springer, 1993.