ELSEVIER

Computer Networks and ISDN Systems 30 (1998) 925–950

COMPUTER NETWORKS and ISDN SYSTEMS

Partial order models for quantitative extensions of LOTOS

Ed Brinksma [a,1], Joost-Pieter Katoen [b,2], Rom Langerak [a,3], Diego Latella [c,4]

[a] *Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*
[b] *Lehrstuhl für Informatik VII, Friedrich-Alexander University of Erlangen-Nürnberg, Martensstrasse 3, 91058 Erlangen, Germany*
[c] *CNUCE Istituto del CNR, Via Santa Maria 36, 56100 Pisa, Italy*

Abstract

Event structures are a prominent model for non-interleaving concurrency. The use of event structures for providing a compositional non-interleaving semantics to LOTOS without data is studied. In particular, several quantitative extensions of event structures are proposed that incorporate notions like time – both of deterministic and stochastic nature – and probability. The suitability of these models for giving a non-interleaving semantics to a timed, stochastic and probabilistic extension of LOTOS is investigated. Consistency between the event structure semantics and an (event-based) operational semantics is addressed for the different quantitative variants of LOTOS and is worked out for the timed case in more detail. These consistency results facilitate the coherent use of an interleaving and a non-interleaving semantic view in a single design trajectory and provide a justification for the event structure semantics. As a running example an infinite buffer is used in which gradually timing constraints on latency and rates of accepting and producing data and the probability of loss of messages are incorporated.

Keywords: Deterministic time; Event structures; Probability; Process algebra; Semantics; Stochastic time; True concurrency

1. Introduction

Nowadays, it is widely recognised that the system design process should be supported by formal methods and tools, especially in the area of system specification and verification [26]. Traditionally, formal methods have concentrated on the functional aspects of systems such as their observable behaviour, control flow and synchronisation as properties in relative time. More recently, formal methods for the representation and analysis of functional aspects in combination with quantitative aspects of system behaviour have come into focus. They allow the specification of the delay of activities or the probability of actual occurrence of activities.

On the one hand, quantitative models are motivated by the need to analyse system aspects that refer to time and probability. Such aspects are abundantly present in the new generations of distributed systems, which incorporate highly complicated functionalities such as, for example, real-time multi-media capabilities. To

[1] E-mail: brinksma@cs.utwente.nl.
[2] E-mail: katoen@informatik.uni-erlangen.de.
[3] E-mail: langerak@cs.utwente.nl.
[4] E-mail: d.latella@cnuce.cnr.it.

0169-7552/98/$19.00
PII: S0169-7552(97)00134-7

support the development of such systems it is necessary to specify and verify real-time quality-of-service properties like desired throughput, latency and jitter. On the other hand, there is a growing need for integrating performance and reliability analysis into traditional formal methods for functional system behaviour. It would be desirable to obtain performance models, such as Markov chains and simulation models, from system specifications in a systematic and (semi-)automatic manner. This would facilitate the assessment of the efficiency and reliability of design alternatives in early stages of the design. Due to the increasing magnitude and complexity of systems the traditional construction of performance models has become a non-trivial task. Bridging the gap between formal methods and performance analysis is expected to enable the exploitation of the structure of system specifications in an effective way. In addition, the size of derived performance models could be reduced by suitable transformations that are applied already at the specification level.

All together this gives rise to the need for specification languages with formal semantics that support the representation of quantitative aspects. This paper concentrates on process algebras as specification formalism and non-interleaving models – event structures – as semantic foundation.

Process algebra languages (e.g. CCS [61], CSP [43] and ACP [5]) are characterised by the presence of a number of powerful composition operators that facilitate the development of well-structured specifications. Some example composition operators are action-prefix (denoted $;$), choice ($\Box$) and parallel independence ($|||$). In addition, verification and transformation techniques have been rather well investigated in this field. A significant effort has been made in investigating quantitative extensions of process algebras. In this paper we concentrate on the process algebra LOTOS, which is a formal description technique standardised by ISO [44].

The semantics of LOTOS and many other process algebras does not explicitly model the independence of subsystems. Instead, independence is modelled by the arbitrary alternation of independent parallel actions. It is therefore called *interleaving* semantics. For instance,

$$(a;P) ||| (b;Q) \equiv a;(P ||| (b;Q)) \Box b;((a;P) ||| Q), \tag{1}$$

where a,b are actions, P,Q are processes, and $\equiv$ denotes semantic equivalence. This transformation of parallel independence into a combination of prefix and choice is (in its full form) known as the *expansion law*. In interleaving semantics only the global states of a system are considered, i.e. one abstracts from the distribution of a system. This assumption is technically quite convenient and has led to many interesting and useful results. However, when a system is considered at a less abstract level, e.g. when timing or performance aspects play an explicit role, the interleaving assumption may no longer be convenient. At this level of abstraction the intensional system characteristics often dominate design considerations. It becomes important to understand how actions are scheduled in time and with what probability alternative executions can appear, which at a higher level of abstraction could all be faithfully modelled by similar non-deterministic constructions. In such cases *non-interleaving* or *true concurrency* semantics are considered to be an interesting alternative. In these semantic models the expansion law is abandoned and explicit information is retained about the parallelism between system components.

An attractive non-interleaving model for concurrency is formed by the family of *event structures* [81]. Event structures have as their basic ingredient events that are labelled with actions. An event models the occurrence of an action. To fit the specific requirements of LOTOS parallel composition and disabling we use a variant of event structures called *(extended) bundle event structures* [51,52]. This paper introduces this model and presents several quantitative extensions of it. It will be shown how these quantitative extensions can be used to provide a compositional non-interleaving semantics to extensions of LOTOS that incorporate time – both of deterministic and stochastic nature – and probabilities.

In our approach the fundamental constraints which have been a driving force are: compatibility, minimality, and consistency. *Compatibility* entails that the quantitative extensions of event structures should be obtained by adding elements to the plain model without changing the ingredients of the latter. In this way a clear relation can be achieved between the non-interleaving semantics of LOTOS and the semantics of its quantitative successors. *Minimality* implies that the conceptual ingredients of the model as well as the additions to the process algebra

due to the incorporation of quantitative information should be as limited as possible. This keeps the model and the process algebra simple to comprehend. *Consistency* means that there should exist a clear and formal relation between the non-interleaving semantics and an interleaving semantics for each quantitative extension of LOTOS. This not only provides evidence for our event structure semantics, but also facilitates the coherent use of both types of semantic views in the design process.

This paper is aimed to provide a (somewhat quick and informal) tour through – to our opinion – some interesting results that we achieved in our research on extensions of event structures. The main concepts that are needed for a proper understanding of the subject will be formally defined. We shall mainly use examples for illustrating the ideas and will refer for more technical treatments to our more detailed papers.

The organisation of this paper is as follows. Section 2 recapitulates the basic constructs of LOTOS without data and shows that (extended bundle) event structures are a suitable non-interleaving model for it. Section 3 introduces real-time aspects in event structures and considers timed operators such as timeout, watchdog, and timed action-prefix. Section 4 shows how the timed model can be generalised such that the delay of actions is determined in a stochastic way. Section 5 deals with probabilistic aspects that allow one to quantify the likelihood of certain alternatives happening (i.e. probabilistic choice). Each of these sections reports on compatibility with the plain model and consistency with an operational semantics that provides a blue-print for obtaining an interleaving semantics. This is worked out for the timed case in some detail. As a running example an infinite buffer is used in which gradually time constraints on latency and rates of accepting and producing data and the probability of loss of messages are incorporated. Section 6 provides an extensive comparison with existing work and places our work in context of others. Finally, Section 7 concludes with the main results and addresses some future work.

2. Basic LOTOS and event structures

In this section we briefly introduce the process algebra Basic LOTOS and the semantic model of *bundle event structures* which can be used for giving a true concurrency semantics to this algebra. We use *Basic* LOTOS which means that we do not consider data types and value passing.

2.1. Basic LOTOS

Basic LOTOS has a parallel operator that allows multi-party synchronisation (like in CSP). It has a special internal action **i** modelling internal (i.e. unobservable) activity. Another distinctive feature is the so-called *disabling* operator which is quite convenient for protocol specification (e.g. a data-phase can be disabled by a disconnection phase). Let P and Q be Basic LOTOS expressions, L be the universe of observable actions, a be an action from $L \cup \{\mathbf{i}\}$, $A \subseteq L$ be a set of actions, f be a function from L to $L \cup \{\mathbf{i}\}$, and x be a process identifier. Then the syntax of Basic LOTOS is recursively given by Table 1. $P \,[\![\varnothing]\!]\, Q$ is abbreviated as $P \,|||\, Q$.

Table 1
Syntax of Basic LOTOS

inaction	**stop**	disabling	$P [> Q$
successful termination	**exit**	parallel composition	$P \,\|[A]\|\, Q$
action-prefix	$a ; P$	hiding	**hide** A **in** P
choice	$P [] Q$	relabelling	$P[f]$
sequential composition (enabling)	$P >> Q$	process instantiation	x

Table 2
Standard interleaving semantics of Basic LOTOS

$$\frac{}{\mathbf{exit} \xrightarrow{\delta} \mathbf{stop}} \qquad \frac{}{a\,;\,P \xrightarrow{a} P} \qquad \frac{P \xrightarrow{a} P'}{\begin{array}{l} P \,[]\, Q \xrightarrow{a} P' \\ Q \,[]\, P \xrightarrow{a} P' \end{array}}$$

$$\frac{P \xrightarrow{a} P'}{P >> Q \xrightarrow{a} P' >> Q} \quad (a \neq \delta) \qquad \frac{P \xrightarrow{\delta} P'}{P >> Q \xrightarrow{\mathbf{i}} Q} \qquad \frac{P \xrightarrow{a} P'}{\begin{array}{l} P \,[>\, Q \xrightarrow{a} P' \,[>\, Q \\ Q \,[>\, P \xrightarrow{a} P' \end{array}} \quad (a \neq \delta)$$

$$\frac{P \xrightarrow{\delta} P'}{\begin{array}{l} P \,[>\, Q \xrightarrow{\delta} P' \\ Q \,[>\, P \xrightarrow{\delta} P' \end{array}} \qquad \frac{P \xrightarrow{a} P'}{P[f] \xrightarrow{f(a)} P'[f]} \qquad \frac{P \xrightarrow{a} P'}{x \xrightarrow{a} P'} \quad (x := P)$$

$$\frac{P \xrightarrow{a} P'}{\begin{array}{l} P \,|[A]|\, Q \xrightarrow{a} P' \,|[A]|\, Q \\ Q \,|[A]|\, P \xrightarrow{a} Q \,|[A]|\, P' \end{array}} \quad (a \notin A \cup \{\delta\}) \qquad \frac{P \xrightarrow{a} P', Q \xrightarrow{a} Q'}{P \,|[A]|\, Q \xrightarrow{a} P' \,|[A]|\, Q'} \quad (a \in A \cup \{\delta\})$$

$$\frac{P \xrightarrow{a} P'}{\mathbf{hide}\ A\ \mathbf{in}\ P \xrightarrow{a} \mathbf{hide}\ A\ \mathbf{in}\ P'} \quad (a \notin A) \qquad \frac{P \xrightarrow{a} P'}{\mathbf{hide}\ A\ \mathbf{in}\ P \xrightarrow{\mathbf{i}} \mathbf{hide}\ A\ \mathbf{in}\ P'} \quad (a \in A)$$

The syntax of a process definition is $x := P$. A set of process definitions is called a *process environment*. A behaviour expression is always considered in the context of a process environment. The semantics of Basic LOTOS is given by the derivation rules in Table 2. Here, the special action δ models the successful termination of an expression. By Table 2 a transition system is defined for each Basic LOTOS expression. Often transition systems are too concrete in the sense that we would like to identify certain behaviour expressions with different transition systems, e.g. on the basis of observability criteria. For these reasons many equivalences and pre-orders have been defined; for an overview see Refs. [28,29].

2.2. Event structures

An attractive non-interleaving model for concurrency is formed by the family of *event structure* models. Event structures have as their basic ingredient events labelled with actions; an event models the occurrence of its action. Different events can have the same action label, implying that they model different occurrences of the action. We are in general not really interested in the event identities as such (so implicitly we work modulo an event renaming morphism), as the events just serve to identify or distinguish action occurrences. Often we will be sloppy and denote an event by its action label, if no confusion arises.

Several relations can exist between the events in a system. Two events are said to be in *conflict* if there is no system run in which both events happen. The fact that certain events have to happen before other events is captured by some kind of *causality* relation. Different event structure models differ in the way causality is modelled, see Section 6.

In bundle event structures [51,52], causality is represented by *bundles*: a bundle is a pair (X,e) with X a set of events and e an event. The set of all bundles is denoted by $\mapsto$ and we denote a bundle (X,e) by $X \mapsto e$. The meaning of a bundle $X \mapsto e$ is that X is a set of causal conditions for e, in the sense that if e happens, one of the events in X has to have happened before. If several bundles point to e, for each bundle set an event should have happened. In addition, we demand that for each bundle $X \mapsto e$, all the events in X are in mutual conflict with each other such that at most one event in X can happen. In this way, if e has happened, exactly one event from X has happened before, so there is no doubt about the causal predecessors of e.

Most event structure models use a symmetric binary conflict relation, denoted by #. In bundle event structures we use a conflict relation that is not necessarily symmetric, denoted by $\rightsquigarrow$. The interpretation of $e \rightsquigarrow e'$ is the following: if e and e' both occur in a system, then e has to occur before e'. This notion is very convenient for modelling the LOTOS disabling operator (see Fig. 2). Note that if $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$ then this means that in fact e and e' cannot occur together in a system run; for this reason we denote the symmetric part of $\rightsquigarrow$ by the conflict symbol #, i.e. $e \# e'$ if and only if $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$.

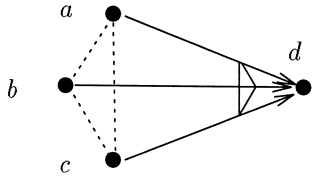
Bundle event structures with $\rightsquigarrow$ were called *extended* bundle event structures in Refs. [51,52]; for brevity we will just call them event structures in this paper.

Definition 1 (*Event structure*). An *event structure* $\mathcal{E}$ is a quadruple $(E, \rightsquigarrow, \mapsto, l)$ with:
- E, a set of *events*,
- $\rightsquigarrow \subseteq E \times E$, the irreflexive *conflict* relation,
- $\mapsto \subseteq \mathcal{P}(E) \times E$, the *bundle* relation,
- $l : E \to L \cup \{ \mathbf{i} \}$, the *labelling* function,

such that $X \mapsto e \Rightarrow \forall e_1, e_2 \in X : (e_1 \neq e_2 \Rightarrow e_1 \rightsquigarrow e_2)$.

We represent an event structure graphically in the following way. Events are drawn as dots; near the dot we sometimes give the event name and/or the action. Conflicts are indicated by dotted arrows; if the conflict is symmetric we draw it by a dotted line. A bundle $X \mapsto e$ is indicated by drawing an arrow from each element of X to e and connecting all the arrows by small lines.

The following picture is an example of an event structure, with a bundle $\{a, b, c\} \mapsto d$:



The bundle here means that for d to happen, one of a, b or c should have happened already.

Example 2. Consider a first-in first-out (FIFO) buffer of infinite capacity. Let wr_n denote the nth writing (i.e. insertion) in the buffer and rd_n denote the nth reading (i.e. removal) from the buffer. An event structure that models such a FIFO buffer is depicted in Fig. 1. Remark that if the buffer is non-empty, reading and writing are to a certain extent independent. A usual interleaving representation of this buffer would prescribe a mutual exclusion between reading and writing, which is unnatural when for example considering the buffer as a communication channel where reading and writing take place at different places.

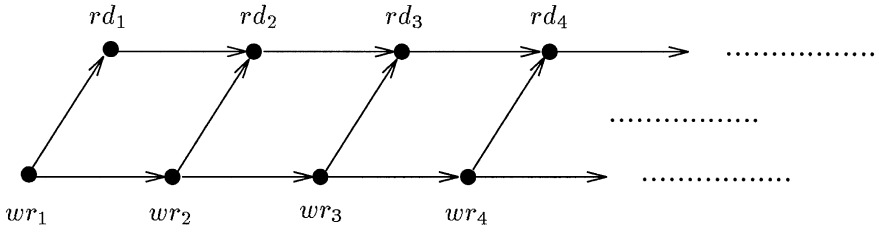


Fig. 1. Event structure representation of a FIFO buffer with infinite capacity.

The concept of a system run for an event structure is captured by the notion of an *event trace*, which is a sequence of events, where each event is preceded by its causal predecessors, and the $\rightsquigarrow$ relation is respected:

Definition 3 (*Event trace*). Let $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$ be an event structure. An *event trace* is a sequence $e_1 \ldots e_n$ of distinct events with $e_1, \ldots, e_n \in E$, satisfying for all $0 < i \leqslant n$:
1. $\forall j : e_i \rightsquigarrow e_j \Rightarrow i < j$,
2. $\forall X \subseteqq E : X \mapsto e_i \Rightarrow \{e_1, \ldots, e_{i-1}\} \cap X \neq \emptyset$.

With the help of event traces we can define a semantics for event structures in terms of (labelled) partial orders [51], which form a natural and attractive basic semantics for comparing true concurrency models [72].

2.3. Non-interleaving semantics for Basic LOTOS

Event structures can be used for giving a compositional semantics to Basic LOTOS. Recursion is handled by defining a cpo (complete partial order) on event structures and then applying standard fixed point theory. We refer to Refs. [51,52] for technical details and just give some examples in Fig. 2.

The derivation rules for Basic LOTOS given in Table 2 can be adapted in such a way as to include event information. Let each occurrence of an action-prefix and **exit** be superscripted with an arbitrary but unique event identifier, denoted by a Greek letter. These identifiers play the role of event names. For parallel composition new event names can be created. If $e \in P$ and $e' \in Q$, then possible new names for events in $P \,[\![A]\!]\, Q$ are $(e, *)$ and $(*, e')$ for unsynchronised events and (e, e') for synchronised ones. For example, some modified inference rules are:

$$\frac{}{\mathbf{exit}^{\xi} \xrightarrow{\xi,\delta} \mathbf{stop}} \qquad \frac{}{a^{\xi}\ ;\ P \xrightarrow{\xi,a} P} \qquad \frac{P \xrightarrow{\xi,a} P',\ Q \xrightarrow{\psi,a} Q'}{P\,|[A]|\,Q \xrightarrow{(\xi,\psi),a} P'\,|[A]|\,Q'} \quad (a \in A \cup \{\delta\})$$

The set of modified rules constitute an *event-based* operational semantics. Using this semantics we are able to derive exactly the same event traces as obtained from the (denotational) event structure semantics. This entails that the two semantic views are event-trace equivalent. We like to emphasise that this is a rather interesting notion of equivalence in our setting: it coincides with bisimulation equivalence (based on events rather than actions) since the transition system induced by $\rightarrow$ is deterministic, and is equivalent to partial-order equivalence since event traces are as expressive as partial orders [51,52]. Since the interleaving semantics of

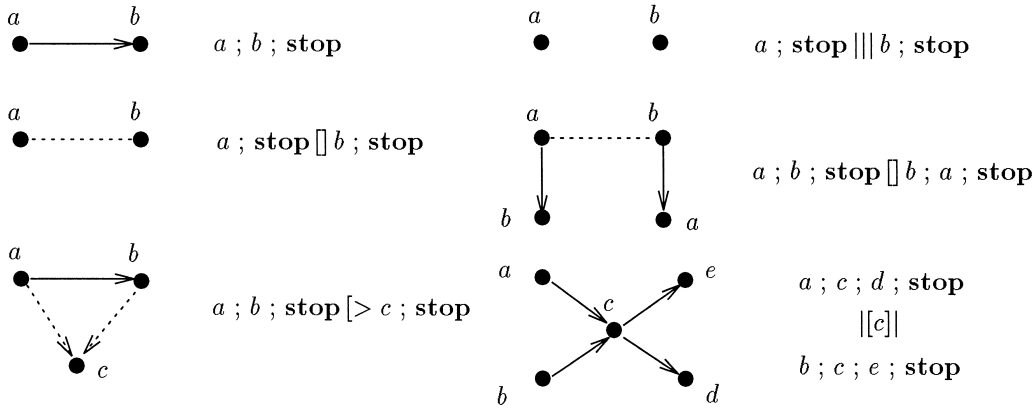


Fig. 2. Event structure semantics for some example Basic LOTOS expressions.

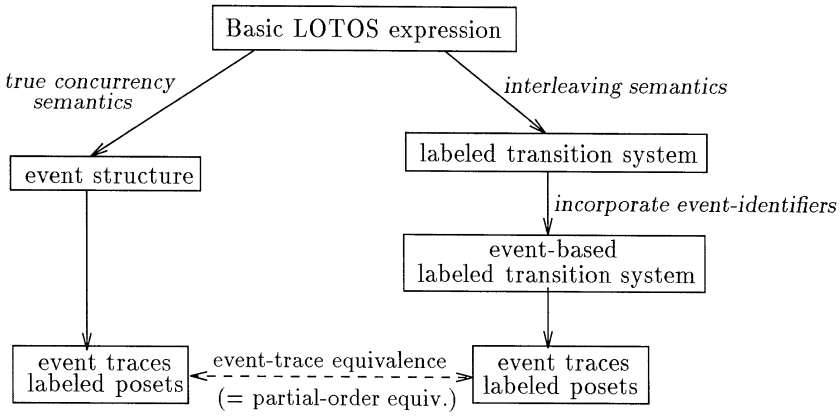


Fig. 3. Consistency between non-interleaving and interleaving semantics.

Basic LOTOS can easily be obtained by simply omitting the event identifiers, these results determine the consistency between the event structure and interleaving semantics. A pictorial representation of this notion of consistency is given in Fig. 3.

3. Adding time to event structures

Timed extensions of process algebras have been investigated thoroughly in the last decade. There are many different ways in which the concept of time can be added to a process algebra and this has resulted in various proposals. It is not the intention of this paper to propose yet another timed variant of Basic LOTOS, but we rather want to show how some elementary operators are modelled in a rather intuitive way using a timed extension of event structures.

We first introduce a timed version of action-prefix and a timed extension of event structures. Then we will show how this model can be used to provide a non-interleaving semantics and we construct a consistent timed operational semantics. Finally, we show how this approach can be extended by incorporating timeout and watchdog operators.

3.1. A simple operator: timed action-prefix

The basic timing construct considered here is *timed action-prefix*. Let $\mathsf{Time} = \mathbb{R}^+ \cup \{0,\infty\}$ be our time domain, T an element of $\mathscr{P}(\mathsf{Time})$, and t an element of Time. For notational convenience, sets $[t,\infty)$ are abbreviated as t and delays equal to $[0,\infty)$ are omitted. The process $a_T\,;P$ is prepared to engage in the action a at any time $t \in T$, where t is measured relative to the time at which the process $a_T\,;P$ was enabled. An interaction can only occur when all participants are ready to engage in it. For instance, consider

$$(a\,; b_{T_1}\,; \mathbf{stop}) \,|[a,b]|\, (a\,; b_{T_2}\,; \mathbf{stop}).$$

If t_a denotes the time of occurrence of action a, action b is can appear at any time in $(t_a + T_1) \cap (t_a + T_2) = t_a + (T_1 \cap T_2)$, where $t + T$ denotes $\{t + t' \mid t' \in T\}$. Interactions may become impossible due to incompatible timing constraints in the participating behaviours, e.g. if $T_1 \cap T_2 = \emptyset$, action b can never occur.

Note that with this simple operator there is not a straightforward translation of the expansion law (1) to the timed case, e.g.,

$$(a_3;P)\,|||\,(b_7;Q) \not\equiv a_3;(P\,|||\,b_7;Q)\,\Box\,b_7;(a_3;P\,|||\,Q).$$

A possible way out would be to allow negative delays and recalculate relative delays as in

$$(a_3;P)\,|||\,(b_7;Q) \equiv a_3;(P\,|||\,b_4;Q)\,\Box\,b_7;(a_{-4};P\,|||\,Q).$$

This would, however, allow $(b,7)(a,3)$ as trace which is usually banned since time goes backwards. In particular, such ill-timed traces are ruled out in timed interleaving semantics since parallel components typically need to synchronise on the passage of time. We will see that in non-interleaving semantics such ill-timed traces are a rather natural phenomenon as long as they respect causality (being succinctly phrased in Ref. [1] as ''ill-timed but well-caused''). This is due to the absence of any mechanism to force time to advance in this model, which enables independent components to evolve independently (rather than forcing them to synchronise on the advance of time). Indeed, Ref. [1] shows that an expansion law can be obtained when ill-timed traces are considered.

3.2. Timed event structures

To specify the relative delay between causally dependent events time is associated with bundles. In order to facilitate the specification of timing constraints on events that have no causal predecessors (i.e. the initial events), time is also associated with events. Though it seems sufficient to only have time labels for initial events, synchronisation of events makes it necessary to allow for equipping all events with time labels, including the non-initial ones.

Definition 4 (*Timed event structure*). A *timed event structure* Γ is a triple $\langle \mathscr{E}, \mathscr{A}, \mathscr{R} \rangle$ with an event structure $\mathscr{E} = (E, \rightsquigarrow, \mapsto, l)$, an *event delay* function $\mathscr{A}: E \to \mathscr{P}(\mathsf{Time})$ and a *bundle delay* function $\mathscr{R}: \mapsto \to \mathscr{P}(\mathsf{Time})$.

Functions $\mathscr{A}$ (for absolute) and $\mathscr{R}$ (for relative) associate a (possibly empty) set of time instants to events and bundles respectively. A bundle $X \mapsto e$ with $\mathscr{R}((X,e)) = T$ is denoted by $X \overset{T}{\mapsto} e$; its interpretation is that if an event in X has happened at a certain time, then e can happen t time units later, for any $t \in T$. $\mathscr{A}(e) = T$ means that e can happen at any $t \in T$ from the beginning of the system, usually time 0.

The intuitive interpretation of timed event structures is explained by means of some small examples, see Fig. 4. Event and bundle delays are denoted near the event and bundle respectively. Like for the syntax, delays $[t,\infty)$ are denoted by t and delays equal to 0, i.e. $[0,\infty)$ are omitted. Event a in Fig. 4a may happen at any $t_a \in \{2,4,6\}$, assuming that executions start at time 0. Event b in Fig. 4b may happen at any $t_b \in t_a + [7,\infty)$, where t_a denotes the time at which a occurred. In the last example, Fig. 4c we put as constraint $t_a \leqslant t_b$ in case both a and b happen (so, a caused b). This corresponds to the common sense idea that causes should occur before their effects.

We now come to the following definition of timed event trace. The basic principle is that an event may happen once all its timing (and, of course, causal) constraints are fulfilled.

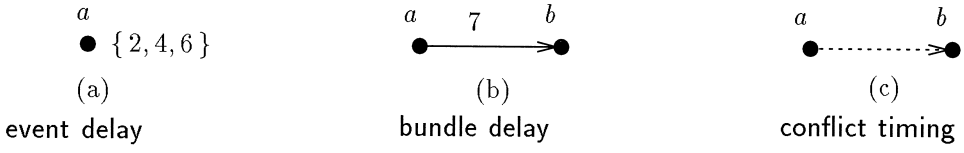


Fig. 4. Forms of induced time constraints.

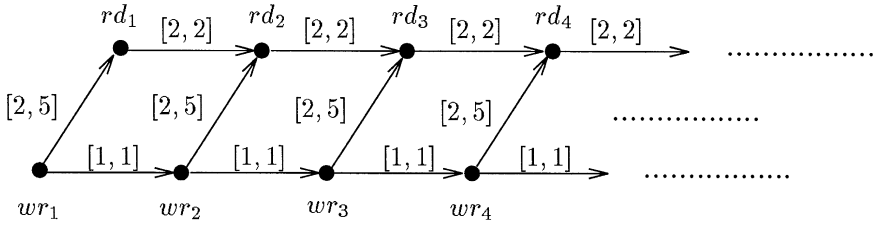


Fig. 5. An event structure representation of a time-constrained FIFO buffer.

Definition 5 (*Timed event trace*). $(e_1,t_1)\ldots(e_n,t_n)$ with $e_i \in E$ and $t_i \in \mathsf{Time}\setminus\{\infty\}$, is a *timed event trace* of $\langle \mathscr{E},\mathscr{A},\mathscr{R}\rangle$ if and only if for all $0 < i \leqslant n$:

1. $e_1 \ldots e_n$ is an event trace of $\mathscr{E}$,
2. $t_i \in \mathscr{A}(e_i)$,
3. $\forall X \subseteqq E : \left(X \overset{T}{\mapsto} e_i \ \wedge\ X \cap \{ e_1,\ldots,e_{i-1}\} = \{ e_j \}\right) \Rightarrow\ t_i \in t_j + T$,
4. $\forall j : e_j \rightsquigarrow e_i \ \Rightarrow\ t_j \leqslant t_i$.

Let $\mathsf{Traces}(\Gamma)$ denote the set of timed event traces of Γ. The first constraint requires a form of backwards compatibility with the untimed model: if all timings are deleted from σ an event trace of $\mathscr{E}$ should result. The second through fourth constraint take care of event delays, bundle delays and timings due to $\rightsquigarrow$ respectively.

Example 6. In this example we treat a time-constrained FIFO buffer. This example is taken from [84]; the only difference is that we consider a buffer of infinite length. The FIFO buffer models a communication network with the following timing constraints: (i) message latency in the range of 2 to 5 time units; (ii) a message input rate of 1 message per time unit; (iii) a message output rate of 1 message per two time units. A timed variation of the FIFO buffer of Fig. 1 that corresponds to these constraints is depicted in Fig. 5.

Timed event traces do respect causality, but not necessarily time. That is, two (or more) independent events can occur in a trace in either order regardless of their timing. For example, $(wr_1,0)(wr_2,1)(wr_3,2)(wr_4,3)(rd_1,2)$ is a timed event trace of the time-constrained FIFO buffer. The ill-timedness is due to the possible interleavings of causally independent events. Since the causal ordering between events implies their temporal ordering the causal ordering can never contradict the temporal order. The following result implies that for any ill-timed event trace σ there exists a time-consistent event trace that can be obtained from σ by swapping repeatedly ill-timed pairs of timed events.

Theorem 7. $\forall\, t' < t : \sigma(e,t)(e',t')\sigma' \in \mathsf{Traces}(\Gamma) \ \Rightarrow\ \sigma(e',t')(e,t)\sigma' \in \mathsf{Traces}(\Gamma)$.

Note that the reverse implication does not hold; for instance, if e causally depends on e' then the order of events $e'\,e$ in a trace cannot be reversed since this would contradict their causal ordering. Due to this theorem it can be proven that two timed event structures that have identical timed event traces also have identical labelled partial orders.

3.3. A non-interleaving timed semantics

The model of timed event structures can be used to provide a non-interleaving denotational semantics to LOTOS for which action-prefix is equipped with a set of time instants. Here, we present this semantics by means of example; a complete definition can be found in Ref. [50].

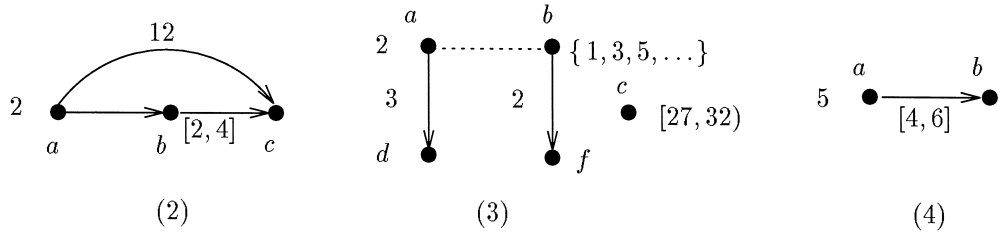


Fig. 6. Timed event structure semantics for some example expressions.

Example 8. Fig. 6 shows the timed event structures corresponding to:

$$a_2\,;\big(b\,;c_{[2,4]}\,;\mathbf{stop}\,|[\,c\,]|\,c_{12}\,;\mathbf{stop}\big), \tag{2}$$

$$\big(a_2\,;d_3\,;\mathbf{stop}\,\Box\,b_{\{1,3,5,\ldots\}}\,;f_2\,;\mathbf{stop}\big)\,|||\,c_{[27,32)}\,;\mathbf{stop}, \tag{3}$$

$$a_2\,;b_{[4,7]}\,;\mathbf{stop}\,|[\,a,b\,]|\,a_5\,;b_{[2,6]}\,;\mathbf{stop}. \tag{4}$$

Case (2) illustrates the need to introduce a bundle from a to c for the sole purpose of making the event delay of c relative to the prefixed event a. Bundle $\{a\}\mapsto b$ is present since b causally depends on a. Case (3) shows how choice and parallel composition without synchronisation are modelled. Case (4) shows that for parallel composition delays may have to be recomputed: intersection of event delays and bundle delays respectively do the job.

3.4. Timed event-based operational semantics

In a similar way as for the untimed case we construct an event-based operational semantics and investigate its correspondence to the timed non-interleaving semantics. Since most timed process algebras are based on an (operational) interleaving semantics this also facilitates a comparison with existing approaches. The operational semantics defines a set of transition relations $\xrightarrow{(e,a,t)}$. $P\xrightarrow{(e,a,t)}Q$ denotes that P can perform event e, labelled with action a, at time t $(t\neq\infty)$, and subsequently evolve into Q. $\rightarrow$ is the smallest relation closed under all inference rules of Table 3, taken from [46].

Two auxiliary semantic operators are used. ${}^t[\,P\,]$ can be considered as behaviour P shifted t time units in advance. ${}^t\{\,P\,\}$ behaves like P except that it is unable to perform events before t. Nested occurrences of ${}^t[\,]$ and ${}^t\{\,\}$ can be simplified by: ${}^t[\,{}^{t'}[\,P\,]\,]={}^{t+t'}[\,P\,]$ and ${}^t\{\,{}^{t'}\{\,P\,\}\,\}={}^{max(t,t')}\{\,P\,\}$.

The rules for process instantiation are somewhat unusual since we deal with events rather than actions. It is assumed that each process instantiation of x is uniquely identified. Different occurrences of the same process instantiation should produce different event transitions. In addition, event transitions cannot be repeated. For

$$x:=a^{\xi}_{[2,7]}\,;x_{\pi}$$

we first have a transition with (ξ,a,t) for $t\in[2,7]$; the next time that action a occurs it should be labelled with a label different from ξ. This is achieved by using an event renaming operator that prefixes all events in a process with a certain occurrence identifier. $\pi(x)$ is process x where all event identifiers in x are prefixed with π. So, we have

$$\frac{Q\xrightarrow{(\xi,a,t)}Q'}{x_{\pi}\xrightarrow{(\pi\xi,a,t)}\pi(Q')}\qquad(x:=Q)\qquad\frac{P\xrightarrow{(\xi,a,t)}P'}{\pi(P)\xrightarrow{(\pi\xi,a,t)}\pi(P')}$$

This concludes the (timed) event-based operational semantics. The consistency between this semantics and the denotational semantics in terms of timed event structures is similar to the untimed case: timed event-trace

Table 3
Event-based operational semantics for timed Basic LOTOS

$$\frac{}{\mathbf{exit}^{\xi} \xrightarrow{(\xi,\delta,t)} \mathbf{stop}} \qquad \frac{}{a_T^{\xi};\, P \xrightarrow{(\xi,a,t)} {}^{t}[P]} \quad (t \in T)$$

$$\frac{P \xrightarrow{(\xi,a,t)} P'}{{}^{t'}[P] \xrightarrow{(\xi,a,t+t')} {}^{t'}[P']} \qquad \frac{P \xrightarrow{(\xi,a,t)} P'}{\begin{array}{l} P \,[]\, Q \xrightarrow{(\xi,a,t)} P' \\ Q \,[]\, P \xrightarrow{(\xi,a,t)} P' \end{array}}$$

$$\frac{P \xrightarrow{(\xi,a,t)} P'}{P >> Q \xrightarrow{(\xi,a,t)} P' >> Q} \quad (a \neq \delta) \qquad \frac{P \xrightarrow{(\xi,\delta,t)} P'}{P >> Q \xrightarrow{(\xi,\mathbf{i},t)} {}^{t}[Q]}$$

$$\frac{P \xrightarrow{(\xi,a,t)} P'}{\begin{array}{l} P \,[>\, Q \xrightarrow{(\xi,a,t)} P' \,[>\, {}^{t}\{Q\} \\ Q \,[>\, P \xrightarrow{(\xi,a,t)} P' \end{array}} \quad (a \neq \delta) \qquad \frac{P \xrightarrow{(\xi,\delta,t)} P'}{\begin{array}{l} P \,[>\, Q \xrightarrow{(\xi,\delta,t)} P' \\ Q \,[>\, P \xrightarrow{(\xi,\delta,t)} P' \end{array}}$$

$$\frac{P \xrightarrow{(\xi,a,t)} P'}{P[f] \xrightarrow{(\xi,f(a),t)} P'[f]} \qquad \frac{P \xrightarrow{(\xi,a,t)} P'}{{}^{t'}\{P\} \xrightarrow{(\xi,a,t)} {}^{t'}\{P'\}} \quad (t \geqslant t')$$

$$\frac{P \xrightarrow{(\xi,a,t)} P'}{\begin{array}{l} P \,|[A]|\, Q \xrightarrow{((\xi,*),a,t)} P' \,|[A]|\, Q \\ Q \,|[A]|\, P \xrightarrow{((*,\xi),a,t)} Q \,|[A]|\, P' \end{array}} \quad (a \notin A \cup \{\delta\}) \qquad \frac{P \xrightarrow{(\xi,a,t)} P',\ Q \xrightarrow{(\psi,a,t)} Q'}{P \,|[A]|\, Q \xrightarrow{((\xi,\psi),a,t)} P' \,|[A]|\, Q'} \quad (a \in A \cup \{\delta\})$$

$$\frac{P \xrightarrow{(\xi,a,t)} P'}{\mathbf{hide}\ A\ \mathbf{in}\ P \xrightarrow{(\xi,a,t)} \mathbf{hide}\ A\ \mathbf{in}\ P'} \quad (a \notin A) \qquad \frac{P \xrightarrow{(\xi,a,t)} P'}{\mathbf{hide}\ A\ \mathbf{in}\ P \xrightarrow{(\xi,\mathbf{i},t)} \mathbf{hide}\ A\ \mathbf{in}\ P'} \quad (a \in A)$$

equivalence. Due to determinism this coincides with timed bisimulation equivalence (based on events). In addition, due to Theorem 7 timed event-trace equivalence induces (timed) partial-order equivalence.

3.5. Other timed operators

The timed language considered so far is quite simple and is not sufficient to specify realistic real-time systems. We will now consider how two useful timed operators, a watchdog (i.e. timed interrupt) and timeout operator can be incorporated in our approach.

$P \overset{t}{\blacktriangleright} Q$ behaves like P, but at time t control is passed to Q, provided P has not successfully terminated. As a basis for the event structure semantics of this construct we take $P\,[>\,Q$. Then we ensure that all events in P can only occur up to time t (by intersecting event delays with $[0,t]$) and postpone all events in Q by t. An example is shown in Fig. 7. Let P (see Fig. 7a) be the expression

$$a_5\,;\, b_{[4,9]}\,;\, \mathbf{stop} \;|||\; c_{[\,27,32)}\,;\, \mathbf{stop}$$

and Q (see Fig. 7b) the expression $f_2\,;\, g_{[2,4]}\,;\, \mathbf{stop}$. Then Fig. 7c shows the semantics of $P\,[>\,Q$, and Fig. 7d shows $P \overset{6}{\blacktriangleright} Q$. Due to the presence of the non-symmetric conflict relation in our event structures the watchdog

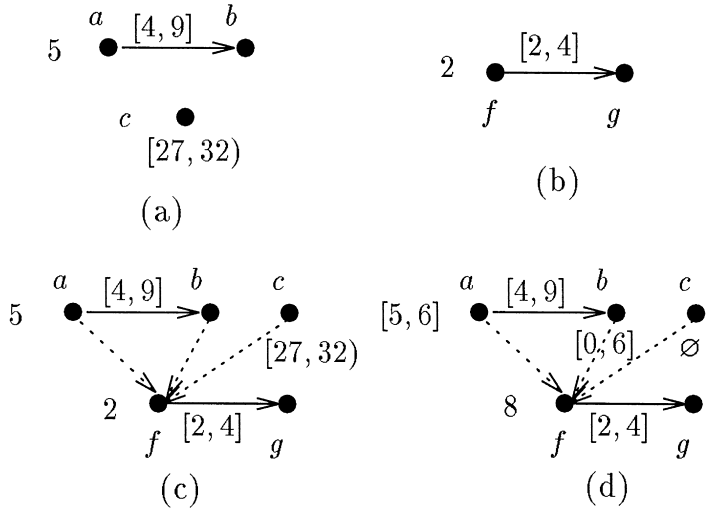


Fig. 7. Event structure representation of timed interrupt.

operator can be modelled without any modification to our model. The inference rules of the watchdog operator are:

$$\frac{P \xrightarrow{(\xi,\delta,t')} P'}{P \overset{t}{\blacktriangleright} Q \xrightarrow{(\xi,\delta,t')} P'}\ (t' \leqslant t) \qquad \frac{P \xrightarrow{(\xi,a,t')} P'}{P \overset{t}{\blacktriangleright} Q \xrightarrow{(\xi,a,t')} P' \overset{t}{\blacktriangleright} Q}\ (a \neq \delta,\ t' \leqslant t) \qquad \frac{Q \xrightarrow{(\xi,a,t')} Q'}{P \overset{t}{\blacktriangleright} Q \xrightarrow{(\xi,a,t+t')} {}^{t}[Q']}$$

The timeout expression $P \overset{t}{\triangleright} Q$ behaves initially like P, but if P has not performed an action before time t control is passed to Q. Up till now, once enabled an event may happen at a certain specified point of time, but it is not forced to happen. In order to faithfully model the principle of a timeout we need a new kind of event: *urgent* events (which we will denote as open dots). Urgent events are forced to occur once they are enabled. The event structure semantics of $P \overset{t}{\triangleright} Q$ is constructed as follows. A new urgent internal event is introduced with delay $[t,t]$ that models the expiration of a timer. Since either the timer expires or P performs an initial action before (or at) t, the timeout event is put in mutual conflict with all initial events of P. The events of Q can only occur after the timeout; this is modelled in the same way as for action-prefix.

Example 9. A typical example of the use of timeouts is

$$Send_2 ; \left(Receive_{[3,7)} ; \mathbf{stop} \overset{5}{\triangleright} Abort ; \mathbf{stop} \right).$$

Following the recipe described above, the timed event structure obtained for this process is depicted in Fig. 8. Its interpretation is as follows. If *Send* occurs at time t $(t \geqslant 2)$ then *Receive* is enabled in $[t+3, t+7)$; if

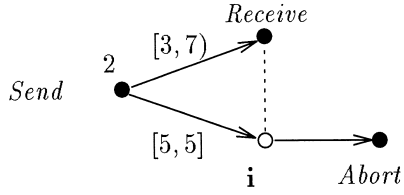


Fig. 8. Modelling timeouts in event structures.

Receive has not occurred at $t+5$, event **i** occurs at that time – modelling a timeout occurrence – and *Abort* becomes immediately enabled.

For the event-based operational semantics we need to keep track of the timeout events. Therefore, we subscript each occurrence of $\triangleright$ with an event identifier, like for action-prefix and **exit**. The presence of urgent events somewhat complicates the inference rules. For instance, for

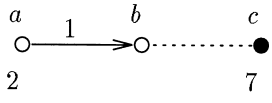
$$a_{12}\,;\mathbf{stop}\,\square\left(b_{18}\,;\mathbf{stop}\overset{5}{\underset{\xi}{\triangleright}}c_{[1,7)}\,;\mathbf{stop}\right)$$

we should not be able to derive that a can occur at time 12, since the timeout event ξ is forced at time 5 and will prevent a from happening. In general, if P performs an event at time t then $P\,\square\,Q$ can perform the same provided that Q is not forced to perform an urgent event at any time earlier. By symmetry, a similar condition is obtained for Q performing an event. Analogous conditions are needed for $[>$, $\triangleright$ (see below) and $\blacktriangleright$. The rules for the other operators remain unaffected. Let $\mathsf{mt}(P)$ denote the minimal time at which P can initially perform an urgent event. Then we obtain for the timeout operator the inference rules:

$$\frac{P\xrightarrow{(\xi,a,t')}P'}{P\overset{t}{\triangleright}_{\psi}Q\xrightarrow{(\xi,a,t')}P'}\quad(t'\leqslant t)\qquad\qquad\frac{}{P\overset{t}{\triangleright}_{\psi}Q\xrightarrow{(\psi,\mathbf{i},t)}{}^{t}[Q]}\quad(t\leqslant\mathsf{mt}(P))$$

The function $\mathsf{mt}(P)$ can easily be defined by induction on the structure of P; we omit the details here and refer to Ref. [50] for further details.

Since urgent events are only used for defining the semantics of $\triangleright$ the introduction of urgent events is rather controlled and their effect can be significantly restricted. This means, for instance, that Theorem 7 also holds in this new setting (implying the coincidence of event-trace equivalence and partial-order equivalence). This is of interest, since in general, urgency has a global impact and the attractiveness of event structures reduces. If we would allow any event to become possibly urgent (for instance, by using the language constructs proposed in Ref. [9]) then we could obtain a structure like



In this structure event c can never happen since a is forced at time 2 and event b subsequently at time 3. Thus, a excludes c from happening though they seem to be completely independent! It appears that the conflict between b and c ''propagates back'' to a conflict between a and c. This aspect implies that the enabledness of an event cannot be simply determined by considering its local causal dependencies and conflicts as for the untimed case. Event structures with this form of urgency are thoroughly considered in Ref. [49].

4. Stochastic timing

Nowadays, performance analysis has a well-recognised position in the design of complex distributed systems. Usually, performance models like queueing networks and Markov chains are developed by abstracting from the system specification that is being used for the qualitative analysis and functional design. In this way, obtaining performance models from system specifications is largely based on human ingenuity and experience. The increasing complexity and magnitude of systems complicates this task considerably. Therefore, it is more and more necessary to obtain performance models in a *compositional* way by exploiting the structure of the system specification at hand. Since process algebras are typically characterised by the presence of a number of powerful

composition operators that facilitate the development of well-structured specifications, these needs resulted in the investigation of *stochastic process algebras* [40,42]. These formalisms are extensions of process algebras, such as CSP, CCS, and LOTOS, where the time of occurrence of actions is determined by (continuous) random variables.

The semantics of stochastic process algebras like PEPA [42], MTIPP [31,39], and EMPA [8] that are restricted to the use of exponential distributions [5] is defined using an extension of labelled transition systems. The structure of transition systems closely resembles the structure of Markov chains, which is an advantage when trying to obtain a performance model directly from the formal specification. In this way the well-known techniques and tools for obtaining for example steady state probabilities for (ergodic) Markov chains can be adopted for performance assessment of the formal specification. In addition, the memoryless property of exponential distributions enables a smooth incorporation of such distributions into an interleaving setting, since

$$(a_\lambda ; P) \,|||\, (b_\mu ; Q) \equiv a_\lambda ; (P \,|||\, b_\mu ; Q) \,\square\, b_\mu ; (a_\lambda ; P \,|||\, Q).$$

Here, a_λ denotes that a occurs after a delay determined by an exponentially distributed random variable with rate λ (and similarly for b_μ). The reason that this law holds is that the time until the occurrence of b after the occurrence of a is distributed according to μ irrespective of how much time has elapsed until a occurred. (By symmetry, a similar reasoning applies when b occurs before a.)

The interleaving of causally independent actions, however, complicates the incorporation of more general (non-memoryless) distributions in transition systems considerably. If we, for instance, generalise the above idea by equipping actions with generally distributed random variables (denoted a_U for random variable U) we obtain that

$$a_U ; P \,|||\, b_V ; Q \not\equiv a_U ; (P \,|||\, b_V ; Q) \,\square\, b_V ; (a_U ; P \,|||\, Q).$$

In case the memoryless property is not satisfied the residual lifetime of V after the occurrence of a must be computed in order to correctly deduce the time until b's occurrence (and, by symmetry, an analogous procedure must be carried out for the residual lifetime of U if b occurs first).

We may thus conclude that exponential distributions and interleaving semantics fit well together, but that general distributions and interleaving semantics do not. From a practical point of view, however, the incorporation of general distributions is considered to be essential in order to faithfully model, for instance, high-speed communication networks and work-flow management systems.

In this section we show how general distributions can be incorporated in event structures. The basic idea is to replace in timed event structures the deterministic delays by random variables. Let RV be a class of continuous (non-negative) random variables.

Definition 10 (*Stochastic event structure*). A *stochastic event structure* is a triple $\langle \mathscr{E}, \mathscr{A}, \mathscr{R} \rangle$ with an event structure $\mathscr{E} = (E, \rightsquigarrow, \mapsto, l)$, an *event distribution* function $\mathscr{A} : E \to \mathsf{RV}$, and a *bundle distribution* function $\mathscr{R} : \mapsto \to \mathsf{RV}$.

(For convenience we use the same notations as for timed event structures.) The interpretation of stochastic event structures is explained by example, see Fig. 9. Like for the timed model, event and bundle distributions are denoted near the event and bundle respectively. We assume the existence of a random variable $\mathbf{0} \in \mathsf{RV}$ which is characterised by $\mathsf{Pr}(\mathbf{0} \leqslant 0) = 1$. We usually omit $\mathbf{0}$ random variables. The enabling time of b in Fig. 9a is modelled by random variable $t_a + U$ where t_a denotes the time of occurrence of a. In Fig. 9b, if a and b

[5] A distribution function F, defined by $F(x) = 1 - e^{-\lambda x}$ for $x \geqslant 0$ and $F(x) = 0$, for $x < 0$, is an exponential distribution with rate λ, $\lambda \in \mathbb{R}^+$. A well-known property of exponential distributions is the memoryless property: let U be a random variable with exponential distribution F_U defined by $F_U(x) = \mathsf{Pr}(U \leqslant x)$. Then for $x, y \geqslant 0$ we have $\mathsf{Pr}(U \leqslant x + y \mid U > y) = \mathsf{Pr}(U \leqslant x)$.

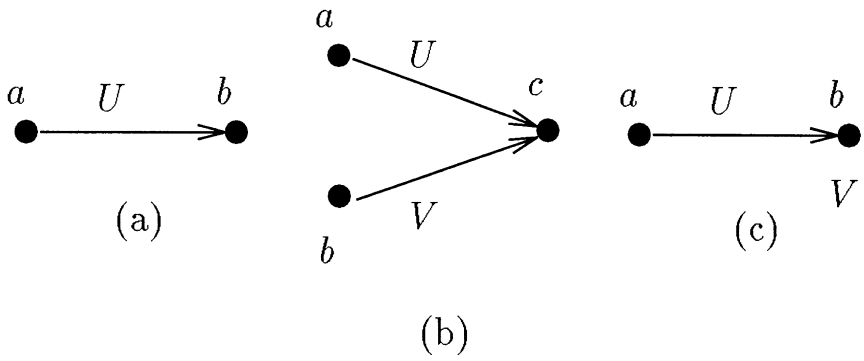


Fig. 9. Some stochastic event structures.

happen at time t_a and t_b respectively, then the enabling time of c is determined by random variable $max(t_a + U, t_b + V)$. This corresponds to the principle that an event can happen once all timing constraints on it have been met. Using a similar reasoning we obtain that the enabling time of event b in Fig. 9c is determined by random variable $max(t_a + U, V)$. For statistical independent random variables U,V with distribution F_U and F_V respectively, $max(U,V)$ has distribution $F_U \cdot F_V$.

The model of stochastic event structures can be used to provide a non-interleaving semantics to LOTOS for which action-prefix is equipped with a continuous random variable from class RV under the condition that RV is closed under *max* and has a unit element for *max* (in our case this is **0**). Exponential distributions do not satisfy this criterion, but, for example, phase-type distributions [63] and distributions of exponential polynomial form [74] do. Both classes of distribution functions include frequently used distributions such as hyper- and hypo-exponential, Erlang and Cox distributions and are widely used in the performance analysis community. The following example illustrates the semantics and justifies the constraints on RV that we mentioned just above. For a full treatment of the semantics and the interpretation of stochastic event structures we refer to Ref. [15].

Example 11. Fig. 10 shows the stochastic event structures corresponding to:

$$a_U \,;\mathbf{stop} \,|[\, a \,]|\, a_V \,;\mathbf{stop}, \tag{5}$$

$$a_U \,;\, b_V \,;\mathbf{stop}, \tag{6}$$

$$(\, a_U \,;\, b_V \,;\mathbf{stop} \,|[\, b \,]|\, b_W \,;\mathbf{stop}) \,|[\, a,b \,]|\, (\, a_X \,;\, b_Y \,;\mathbf{stop} \,|[\, b \,]|\, b_Z \,;\mathbf{stop}). \tag{7}$$

Cases (5) and (7) justify the constraint that RV must be closed under *max*. Case (6) shows the need for having a unit element for *max*: according to our interpretation event b is enabled at a time determined by $max(t_a + U, \mathbf{0})$ which intuitively should correspond to $t_a + U$.

A nice property of labelled transition systems equipped with exponential distributions is that they closely resemble continuous-time Markov chains, see Ref. [40]. Unfortunately, when a non-interleaving semantics is

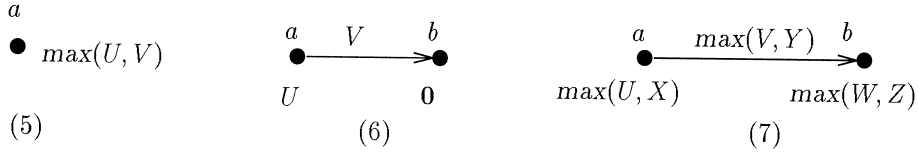


Fig. 10. Stochastic event structure semantics for some example expressions.

adopted the convenient direct relationship between the semantics of stochastic process algebras and Markov-like models is lost. However, the authors have shown in Ref. [47] that it is possible to generate so-called (time-homogeneous) *generalised semi-Markov processes* starting from (a class of) stochastic event structures. GSMPs [77] are a generic model for the study of stochastic discrete-event systems. For such models it is possible to generate simulation runs from which – using the standard simulation techniques for discrete-event systems – performance results can be obtained. This provides a bridge between a formal specification language (a stochastic process algebra) and a performance model (GSMPs).

5. Probabilistic behaviours

In this section we study a probabilistic extension of event structures, called probabilistic event structures. Event structures are equipped with a mechanism for attaching *probabilities* to (internal) events in such a way that groups of these events can suitably model random experiments. Probabilistic event structures will be used as a semantic model for a probabilistic extension of LOTOS. This facilitates the specification of system reliability issues. For instance, the probability of message loss or garbling for an unreliable communication channel or the fault probability of some system component can be specified in such a language.

The events to which probabilities are attached are required to be *internal* so that their occurrence can in no way be affected by the external environment. Consequently, the probability of occurrence of these events can be determined without the need for conditioning probabilities on the possible behaviour of the environment. This is a simplifying assumption which, anyway, still allows the modelling of many interesting situations. In fact, the environment typically has no control over the kind of probabilistic phenomena one is often interested in practice, like the failure of a device [69].

There are in the literature many proposals in which probabilities are associated with *external* events and, in some of them, the environment can also play a role in the resolution of random experiments. In many cases, this leads to more complicated semantic models, since probabilities are conditioned to the actions of the environment and then normalisation is often needed. An overview and classification of such models can be found in Ref. [30].

The probabilistic language construct that we consider is an *internal probabilistic choice*, denoted $\oplus_p$ for $p \in (0,1)$. $P \oplus_p Q$ denotes the process which can internally (i.e. non-deterministically) decide to behave like P or Q. The probability that P will be chosen is p, whereas the probability that Q will be chosen is $1-p$. Probabilistic choices are restricted to be performed between behaviours whose first actions are unobservable. Other forms of non-determinism like $a;P \,[]\, a;Q$ and $a;P \,[]\, \mathbf{i};Q$ are not considered here. This keeps our model as simple as possible. We also notice that

$$a;P \,[]\, a;Q \approx_{te} a;(\mathbf{i};P \,[]\, \mathbf{i};Q),$$

$$a;P \,[]\, \mathbf{i};Q \approx_{te} \mathbf{i};((a;P) \,[]\, Q) \,[]\, \mathbf{i};Q,$$

where $\approx_{te}$ denotes testing congruence [21]. Thus all forms of non-determinism can be rewritten in the required format of our formalism while preserving the notion of testing congruence.

Various probabilistic extensions of process algebras exist in the literature. The semantics of these (asynchronous) process algebras is based on probabilistic extensions of labelled transition systems in which usually a distinction is made between probabilistic and non-probabilistic transitions. The main problem with this approach is the intertwining of these types of transitions. That is to say, it is not clear what the intended meaning is of a probability attached to a transition in the presence of a competitive non-probabilistic transition. Typical behaviours that cause such situations are combinations of parallel composition and probabilistic choice, like

$$(P \oplus_p Q) \,|||\, a;R. \tag{8}$$

The fact that there is one global state in which either a or one of the two probabilistic alternatives can happen makes it difficult to interpret p as the probability that P will be chosen. There have been several solutions

proposed for this problem, but most of them lose the property of backwards compatibility with the non-probabilistic semantics. In this section we will show that a causality-based model, which has no direct notion of global state, does not suffer from these problems.

We extend LOTOS with a probabilistic choice operator $\oplus_p$ and keep the standard choice operator $\square$. Note that $\square$ can model external choice, whereas $\oplus_p$ only models internal choice. To guarantee that $P \oplus_p Q$ induces an independent random experiment some constraints are defined on the syntax. These constraints are intuitively simple but tedious to define formally, see Refs. [46,48]. We illustrate and justify them by means of examples. Each argument P_j of $\oplus$ is required to be either an internal action-prefix $\mathbf{i}; P'_j$ or a probabilistic choice. In the latter case the inner probabilities are conditional to the external choices, e.g.

$$\mathbf{i}; a; \mathbf{stop} \oplus_{0.4} (\mathbf{i}; b; \mathbf{stop} \oplus_{0.2} \mathbf{i}; c; \mathbf{stop})$$

models a random experiment with three possible outcomes, a, b, and c, with probability 0.4, 0.6×0.2 and 0.6×0.8, respectively. On the other hand, expressions like

$$\mathbf{i}; a; \mathbf{stop} \oplus_{0.4} (\mathbf{i}; b; \mathbf{stop} \square \mathbf{i}; c; \mathbf{stop}) \tag{9}$$

are not allowed because the probability of occurrence of b, for example, cannot be determined. The same holds if we replace $\square$ by parallel composition in (9). In addition we do not allow $\oplus_p$ in the context of $\square$ and $[>$. For instance, in

$$a; \mathbf{exit} \,[> (\mathbf{i}; b; \mathbf{stop} \oplus_{0.4} \mathbf{i}; c; \mathbf{stop})$$

the probability of occurrence of b, for instance, cannot be determined since it depends on whether $a; \mathbf{exit}$ terminates successfully or not. The following expression

$$(\mathbf{i}; b; \mathbf{stop} \oplus_{0.4} \mathbf{i}; c; \mathbf{stop}) \,|||\, a; \mathbf{stop}$$

is legal. It denotes the behaviour composed by the stochastic experiment $b; \mathbf{stop} \oplus_{0.4} c; \mathbf{stop}$ and the independent execution of $a; \mathbf{stop}$.

Probabilities are assigned in event structures to internal events grouped into *clusters*. The basic idea is that a cluster represents the possible outcomes of an independent random experiment. The probability $\pi(e) \in (0,1)$ assigned to event e in a cluster models the likelihood that e is the outcome of the experiment given that e is enabled. Events within a cluster mutually exclude each other so that only one event (i.e. the outcome of the experiment) can happen. For purely technical reasons we do not allow clusters to be singletons. This does not pose any practical restriction. Events in a cluster are not in conflict with events outside the cluster; allowing such conflicts would destroy the interpretation that an event probability represents the likelihood that this event happens (once enabled). Finally, all events in a cluster must be pointed to by the same set of bundles. Together with the previous requirement this guarantees that if an event in a cluster is enabled, then *all* events in this cluster are enabled.

Definition 12 (*Cluster*). For event structure $\mathscr{E} = (E, \rightsquigarrow, \mapsto, l)$, set $Q \subseteqq E$ is a *cluster* of $\mathscr{E}$, if and only if $|Q| > 1$ and for all $e, e' \in Q$ and $X \subseteqq E : l(e) = \mathbf{i} \wedge \{ e'' \mid e \rightsquigarrow e'' \} = Q \setminus e \wedge (X \mapsto e \Rightarrow X \mapsto e')$.

A probabilistic event structure is an event structure in which all events of some clusters are assigned a probability. The sum of the probabilities of all events within a cluster should be 1.

Definition 13 (*Probabilistic event structure*). A *probabilistic event structure* is a tuple $\langle \mathscr{E}, \pi \rangle$ with an event structure $\mathscr{E} = (E, \rightsquigarrow, \mapsto, l)$ and a *probability* function $\pi : E \to_p (0,1)$ such that for all $e \in dom(\pi)$:

$$\exists\, Q \subseteqq dom(\pi) : e \in Q \wedge Q \text{ is a cluster} \wedge \sum_{e' \in Q} \pi(e') = 1.$$

$\to_p$ indicates a partial function. Cluster Q can be considered to represent a *random experiment* for which the probability of outcome $e \in Q$ equals $\pi(e)$. The set of event traces of $\langle \mathscr{E}, \pi \rangle$ is simply the set of event

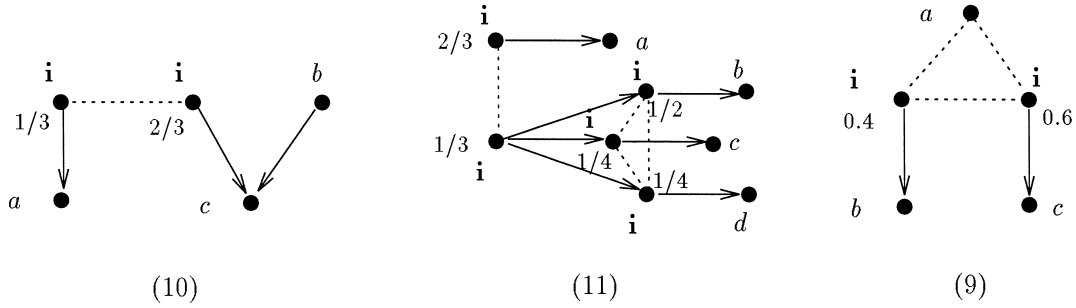


Fig. 11. Some example (non-)probabilistic event structures.

traces of $\mathscr{E}$; the probabilities do not affect the possibility of events to happen, they only quantify the probability of happening.

Example 14. Some example probabilistic event structures are depicted in Fig. 11. Here, the probability of an event is depicted near it. Fig. 11(10) contains a single cluster of 2 events with probabilities $1/3$ and $2/3$ respectively. Fig. 11(11) contains two clusters of two and three events. Fig. 11(9) is not a probabilistic event structure since a is in conflict with events that are assigned a probability.

Example 15. In Fig. 12 a probabilistic version of the FIFO buffer is presented. It models an unreliable channel where messages can be lost with probability $1-p$.

Probabilistic event structures can be used to provide a denotational true concurrency semantics of our probabilistic LOTOS. The cpo semantic domain as well as the interpretation function are straightforward extensions of those for the event structure semantics of Basic LOTOS. Here, we present the semantics by means of example; the complete definition can be found in Ref. [46]. A preliminary version appeared as [48].

Example 16. Fig. 11 shows the probabilistic event structures corresponding to

$$(\mathbf{i}; a; \mathbf{stop} \oplus_{1/3} \mathbf{i}; c; \mathbf{stop}) \,|[c]|\, b; c; \mathbf{stop}, \tag{10}$$

$$\mathbf{i}; a; \mathbf{stop} \oplus_{2/3} \big(\mathbf{i}; b; \mathbf{stop} \oplus_{1/2} (\mathbf{i}; c; \mathbf{stop} \oplus_{1/2} \mathbf{i}; d; \mathbf{stop})\big). \tag{11}$$

Notice that Fig. 11(9) would correspond to the non-allowed expression (9). *(End of example.)*

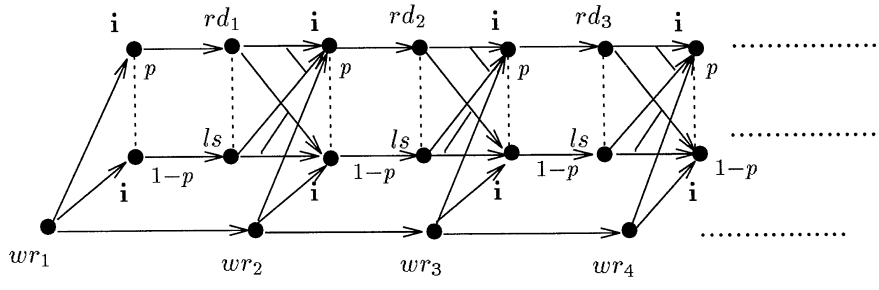


Fig. 12. A FIFO buffer with the possibility of loss of messages.

The relation between the probabilistic non-interleaving semantics and the non-interleaving semantics for Basic LOTOS is as follows. Take a probabilistic LOTOS expression P and consider the non-interleaving semantics of the Basic LOTOS expression obtained by replacing all occurrences of $\oplus_p$ in P by $\Box$. This event structure is *identical* to the one obtained by the probabilistic event structure semantics of P once the probability function π is removed from it. So, the probabilistic case is an conservative extension of the plain case. (This is only rarely the case for probabilistic interleaving semantics.)

In a similar way as for the timed and untimed case, an event-based operational semantics can be defined for probabilistic LOTOS; for details see Chapter 9 of [46]. This semantics uses two inference systems, one for probabilistic and one for non-probabilistic transitions. Probabilistic transitions are labelled by internal events and their probabilities. Moreover they have *priority* over non-probabilistic ones. In this way non-probabilistic transitions cannot be mixed with probabilistic ones: the outgoing transitions of a state are either all non-probabilistic or all probabilistic (in which case their probabilities sum up to 1). This technique solves the problems as induced by e.g. expressions like (8) and is similar to other approaches [36,83].

The consistency between the denotational and operational probabilistic semantics is determined in a slightly different way than depicted in Fig. 3. Construct from a probabilistic event structure a probabilistic transition system – this is rather straightforward by considering event structures as states and transitions as the execution of a single event – and subsequently ignore the probabilities. This (non-probabilistic) transition system is *testing equivalent* [21] with the transition system obtained from removing probabilities from the transition system obtained by the probabilistic operational semantics. Note that this is not such a strong result: in the plain and in the timed case we obtain (timed) bisimulation equivalence. The reason for this weaker result is that in the operational semantics of probabilistic LOTOS, probabilistic transitions have priority over other transitions. In this way, the possibility to perform an observable action may be postponed since probabilistic choices have to be resolved first. This phenomenon is absent in the non-interleaving semantics.

6. Related work

6.1. Partial-order models for LOTOS

A few other partial-order models exist for LOTOS. Ref. [19] uses an operational semantics where each (uniquely identified) action is decorated with a set of causal predecessors. This semantics is a compatible extension of the standard LOTOS semantics and is partial-order equivalent to our semantics. Our event-based operational semantics is a bit simpler since we do not need to record causal predecessors. A denotational semantics in terms of (maximal) partially ordered traces, traces in which each action is enriched with a set of forward pointers to causal successors, is given in Ref. [23]. This results in a linear-time semantics, a semantics in which $a;(b;\mathbf{stop}\,\Box\, c;\mathbf{stop})$ and $a;b;\mathbf{stop}\,\Box\, a;c;\mathbf{stop}$ are semantically equivalent (unlike our approach). Ref. [72] uses labelled posets for providing a semantics to LOTOS. Labelled posets are rather abstract and complex operations are needed for defining a direct semantics for LOTOS.

6.2. Other brands of event structures

The distinguishing features of bundle event structures compared to other event structures are the bundle and non-symmetric conflict relation for modelling parallel composition and disabling respectively in a concise way. (Non-symmetric conflict is also introduced in Ref. [67]). In prime event structures [65] causality is modelled by a partial order. This model is mathematically very elegant and convenient. The drawback is that as a consequence each event has a unique enabling, so if an action can be caused in alternative ways we need different events for the action, harmful to the conciseness of models. In addition it may be rather complicated to define some operations on prime event structures, especially parallel synchronisation [80]. Flow event structures

[10,11] allow alternative enablings but self-conflicting events do pose problems for parallel composition, see Ref. [51]. Stable event structures [81] also allow alternative enablings, but their expressiveness sometimes hampers their analysis, as pointed out by [51]. Extended bundle event structures are more expressive than prime, but are incomparable with flow and stable event structures.

6.3. Time in partial-order models

Timed extensions of partial-order models have received scant attention in the literature. Extensions of pomsets (partially ordered multi-sets) [16], configurations [58], prime event structures [62], posets [45], {and, or}-automata [34] and higher-dimensional automata [33] are known to us, but these models have not been used as a semantic model for a timed process algebra and are merely of theoretical interest. Our approach resembles that of Ref. [24]. [24] proposes a real-time extension of causal trees – equivalence classes of event structures under a non-interleaving notion of bisimulation – and uses this model to provide a semantics to a timed variant of CCS. This approach has later been extended to include time markers that facilitate the specification of relative time delays between arbitrary actions [25]. The main difference with our approach is that we allow for the co-existence of urgent and non-urgent events. Urgency has a subtle influence on causality. For this reason [25] does not consider urgency. We argue that in case urgent events are used for the sole purpose of modelling timeouts, the benefits of event structures can be maintained while retaining a fairly expressive formalism for specifying real-time systems. Recently, Ref. [12] reported on a timed LOTOS extension with a semantics using a timed variant of bundle event structures. Ref. [12] does not include disabling and considers all internal events to be urgent. This leads to similar problems between urgency and causality as reported in Ref. [49].

6.4. Timed process algebras

In the last decade timed extensions of process algebras have received considerable attention. Extensions of languages like ACP [3], CSP [71,75], CCS [36,59,82,38], and LOTOS [9,54,70] have been defined. For an overview of the issues that arise when adding time to process algebras we refer to Ref. [64]. In this paper we showed how the benefits of event structures can be exploited when having a time-prefix specifying at which times an action may occur and using urgency only for modelling timeout scenarios. Most approaches differ in the way urgency is approached. [3] forces each action to happen as soon as possible, [38,75,54], for instance, adopt this strategy for internal actions only, and, in the most extreme case, an operator is introduced that allows any action to be interpreted in an urgent way or not [9]. In [49] we showed that the latter strategy leads to an undesirable connection between causality and urgency. The same applies for the case in which all internal actions occur as soon as possible [13].

6.5. Stochastic process algebras

Stochastic process algebras are a relatively new field compared with timed and probabilistic extensions of process algebras. An introduction and overview can be found in Ref. [40]. Languages like PEPA [42], MTIPP [31,39] and EMPA [8] are restricted to the use of exponential distributions. An interesting result due to Ref. [42] is that (a probabilistic version of) bisimulation has a direct correspondence to lumpability of Markov chains, an important equivalence notion used for reducing the state space of Markov chains. There are only a few stochastic process algebras that allow more general distributions. Ref. [2] define a stochastic extension of LOTOS but require all stochastic timing constraints to be specified at ‘‘top level’’, thus reducing compositionality. Ref. [37] uses a stochastic process algebra to formally describe discrete-event simulation. In their approach non-determinism gives rise to invalid simulations. This is a serious problem since an interleaving semantics is used, i.e., parallelism is ‘‘resolved’’ by non-determinism. This artificial non-determinism is absent in our non-interleaving approach. Other proposals, like [32,68] use decorated transition systems to define a non-inter-

leaving semantics. An approach that is quite close to ours is to use stochastic task graphs as a denotational model [41].

6.6. Probabilistic process algebras

Probabilistic process algebras have been studied quite extensively in the literature. Probabilistic extensions of different process algebras have been proposed, such as ACP [4], CCS (see e.g. Refs. [17,36,83]), CSP [56,57,76], LOTOS [60,66,73,78], and synchronous CCS [27,30,79]. For overviews of probabilistic process algebras we refer to Refs. [18,35]. The models underlying most of these process algebras are labelled transition systems in which probabilities are associated with transitions. To our knowledge our probabilistic LOTOS is the first probabilistic process algebra with a non-interleaving semantics. Several probabilistic (interleaving) process algebras replace the standard choice by a probabilistic one, so that non-determinism is usually modelled by $\oplus_{1/2}$. As a result one has that parallelism also gets implicitly probabilised, i.e.

$$a \,|||\, b \equiv a\,;b \oplus_{1/2} b\,;a.$$

This is taken to the limit in probabilistic ACP and in the proposals for probabilistic LOTOS [66,78] where an explicit probability p can be specified instead of assuming $1/2$ as above. In [4] also the probability of a synchronisation versus an independent move of the components can be specified. The approaches of Refs. [22,36,56,83] are similar to ours in the sense that they do incorporate both a standard and probabilistic choice operator, and besides require probabilistic choices to be independent from the environment. Finally, it is worth noting that the probabilistic extensions proposed in Refs. [4,60,78] are – like in our case – conservative with respect to the plain semantics.

6.7. Consistency of semantics

The consistency between a causality-based and an interleaving semantics has been studied for different languages and models. Our approach is similar to Ref. [11] which relates (amongst others) a flow event structure semantics to the interleaving semantics of CCS. Ref. [6] proves the consistency between an operational semantics for theoretical CSP and a semantics based on labelled prime event structures. Ref. [6] extends the results of Ref. [55] to recursive processes. Ref. [20] proves the consistency between an operational non-interleaving semantics of CCS and denotational semantics based on labelled prime event structures. The notion of consistency between different semantics has been recently studied in a general framework in Ref. [7].

7. Conclusions

Our driving force is to study the expressiveness of partial-order models, in particular event structures, to facilitate formal representation of performance and reliability aspects and to enable performance analysis of formal models in a systematic and (semi-)automated way. In this paper we have presented several quantitative extensions of the formal description technique LOTOS in a non-interleaving setting. Performance analysis is not explicitly treated here and can be found elsewhere [47]. We argued that quantitative extensions are of particular use in a partial-order setting since notions like time, probability and distributions are important at the lower levels of abstraction, where the internal structure of systems is relevant for correct design. Because quantitative extensions of interleaving models abstract away from causality and the distribution of system parts, this makes them less suitable for use at these abstraction levels.

The models presented in this paper are extensions of (extended) bundle event structures, a non-interleaving model tuned to the LOTOS language. For Basic LOTOS it is shown in Refs. [51,52] that this semantic model is consistent with the standard interleaving semantics for LOTOS, thus providing evidence for the adequacy of

bundle event structures. In a similar way (event-based) operational semantics were developed to justify the timed and probabilistic extensions as treated in this paper. A similar result for the stochastic extension restricted to exponential distributions was reported by us in Ref. [15]. In the timed case, this has led to a surprisingly elegant extension of the standard interleaving semantics for LOTOS.

An important reason for this is the absence of any mechanism in the timed event structure model that models the passage of time, which in one way or another makes its appearance in most timed interleaving models. Instead, time is treated as a parameter of the model. This also leads to the absence of time deadlocks, i.e. states in which the passage of time is permanently blocked. If there is an action that cannot be executed (e.g. due to mismatching timing constraints in partners for synchronisation) this leads to the local impossibility of executing the action at hand, but does not have any impact on causally independent parts. It has been shown that these properties are maintained under the addition of watchdog and timeout operators.

An interesting result is that our timed approach can be generalised in a rather straightforward way in order to incorporate probability distribution functions that determine the timing of actions. This caters for the description of more dynamic stochastic behaviour. The model is not restricted to exponential distributions (as opposed to interleaving models) but allows more general distributions such as phase-type distributions to be used. This is essential for the modelling of systems for which the usual Poisson arrival assumptions are no longer reasonable.

Recent developments include the development of algorithms to obtain automatically stochastic simulation models from event structures equipped with distribution functions [47] and the investigation of (plain) event structures in which the constraint that events in a bundle should mutually exclude each other is dropped [53]. Interesting topics for future work are the finite representation of infinite event structures using graph grammars (this would for example facilitate regenerative simulation in the stochastic case), the incorporation of hybrid aspects, the integration of the probabilistic and stochastic extension (an integration of time and probability is considered by us in Ref. [14]), and the definition of equivalences and pre-orders that reflect natural notions of transformation and implementation for timed and stochastic systems.

Acknowledgements

The authors would like to thank Holger Hermanns for comments on a draft version of this paper. The work presented in this paper has been partially funded by C.N.R. - Progetto Bilaterale: Estensioni probabilistiche e temporali dell'algebra di processi LOTOS basate su strutture di eventi, per la specifica e analisi quantitative di sistemi distribuiti, C.N.R. - Progetto Coordinato: Strumenti per la specifica e verifica di proprieta' critiche di sistemi concorrenti e distribuiti, and by the EU as part of the ESPRIT BRA project 6021: Building Correct Reactive Systems (REACT).

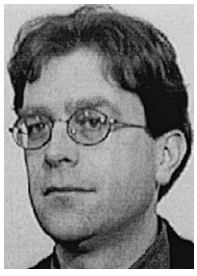
References

[1] L. Aceto, D. Murphy, Timing and causality in process algebra, Acta Inf. 33 (1996) 317–350.
[2] M. Ajmone Marsan, A. Bianco, L. Ciminiera, R. Sisto, A. Valenzano, A LOTOS extension for the performance analysis of distributed systems, IEEE/ACM Trans. Networking 2 (2) (1994) 151–164.
[3] J.C.M. Baeten, J.A. Bergstra, Real time process algebra, Formal Aspects Comput. 3 (2) (1991) 142–188.
[4] J.C.M. Baeten, J.A. Bergstra, S.A. Smolka, Axiomatizing probabilistic processes: ACP with generative probabilities, Inf. Comput. 121 (1995) 234–255.
[5] J.C.M. Baeten, W.P. Weijland, Process Algebra, Cambridge University Press, Cambridge, 1990.
[6] C. Baier, M.E. Majster-Cederbaum, The connection between an event structure semantics and an operational semantics for TCSP, Acta Inf. 31 (1994) 81–104.
[7] C. Baier, M.E. Majster-Cederbaum, How to interpret consistency and establish consistency results for semantics of concurrent programming languages, Fund. Inf. 29 (1997) 225–256.

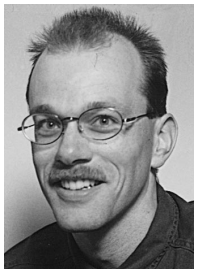
[8] M. Bernardo, R. Gorrieri, Extended Markovian process algebra, in: U. Montanari, V. Sassone (Eds.), Concur'96: Concurrency Theory, LNCS 1119, Springer, Berlin, 1996, pp. 315–330.
[9] T. Bolognesi, F. Lucidi, S. Trigila, Converging towards a timed LOTOS standard, Comput. Standards Interf. 16 (1994) 87–118.
[10] G. Boudol, I. Castellani, Permutations of transitions: An event structure semantics for CCS and SCCS, in: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (Eds.), Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354, Springer, Berlin, 1989, pp. 411–427.
[11] G. Boudol, I. Castellani, Flow models of distributed computations: three equivalent semantics for CCS, Inf. Comput. 114 (1994) 247–314.
[12] H. Bowman, J. Derrick, Extending LOTOS with time; a true concurrency perspective, in: M. Bertran, T. Rus (Eds.), Transformation-Based Reactive Systems Development, LNCS 1231, Springer, Berlin, 1997, pp. 383–400.
[13] H. Bowman and J-P. Katoen, Towards a true concurrency semantics for TE-LOTOS, Technical Report IMMD 7, University of Erlangen-Nürnberg, 1997.
[14] E. Brinksma, J-P. Katoen, R. Langerak, D. Latella, Performance analysis and true concurrency semantics, in: T. Rus, C. Rattray (Eds.), Theories and Experiences for Real-Time System Development, World Scientific, Singapore, 1994, pp. 309–337.
[15] E. Brinksma, J-P. Katoen, R. Langerak, D. Latella, A stochastic causality-based process algebra, Comput. J. 38 (7) (1995) 552–565.
[16] R.T. Casley, R.F. Crew, J. Meseguer, V.R. Pratt, Temporal structures, Math. Struct. Comput. Sci. 1 (2) (1991) 179–213.
[17] I. Christoff, Testing equivalences and fully abstract models for probabilistic processes, in: J.C.M. Baeten and J.-W. Klop (Eds.), Concur'90: Theories of Concurrency – Unification and Extension, LNCS 458, Springer, Berlin, 1990, pp. 126–140.
[18] L. Christoff, Specification and verification models for probabilistic processes, Ph.D. Thesis, Uppsala University, 1993.
[19] J-P. Courtiat, R.J. Coelho da Costa, A true concurrency semantics for LOTOS, in: M. Diaz, R. Groz (Eds.), Formal Description Techniques V, North-Holland, Amsterdam, 1993, pp. 348–362.
[20] P. Degano, R. De Nicola, U. Montanari, On the consistency of 'truly concurrent' operational and denotational semantics (extended abstract), in: 3rd Annual Symp. on Logic in Comput. Sci., IEEE Computer Press, 1988, pp. 133–141.
[21] R. De Nicola, M. Hennessy, Testing equivalences for processes, Th. Comp. Sci. 34 (1984) 83–133.
[22] M. Fang, H.S.M. Zedan, C.J. Ho-Stuart, A model for timed-probabilistic behaviors, J. Syst. Software 28 (1995) 239–251.
[23] C.J. Fidge, Process algebra traces augmented with causal relationships, in: K.R. Parker, G.A. Rose (Eds.), Formal Description Techniques IV, North-Holland, Amsterdam, 1992, pp. 527–541.
[24] C.J. Fidge, A constraint-oriented real-time process calculus, in: M. Diaz, R. Groz (Eds.), Formal Description Techniques V, North-Holland, Amsterdam, 1993, pp. 363–378.
[25] C.J. Fidge, J.J. Žic, A simple, expressive real-time CCS, in: Proc. 2nd Australasian Conf. on Parallel & Real-Time Systems, 1995, pp. 365–372.
[26] S. Gerhart, D. Craigen, T. Ralston, Experience with formal methods in critical systems, IEEE Software (1994) 21–28.
[27] A. Giacalone, C.-C. Jou, S.A. Smolka, Algebraic reasoning for probabilistic concurrent systems, in: M. Broy, C.B. Jones (Eds.), Proc. Working Conf. on Programming Concepts and Methods, North-Holland, Amsterdam, 1990, pp. 443–458.
[28] R.J. van Glabbeek, The linear time – branching time spectrum, in: J.C.M. Baeten and J.-W. Klop (Eds.), Concur'90: Theories of Concurrency – Unification and Extension, LNCS 458, Springer, Berlin, 1990, pp. 278–297.
[29] R.J. van Glabbeek, The linear time – branching time spectrum II (The semantics of sequential systems with silent moves), in: E. Best (Ed.), Concur'93: Concurrency Theory, LNCS 715, Springer, Berlin, 1993, pp. 66–81.
[30] R.J. van Glabbeek, S.A. Smolka, B. Steffen, Reactive, generative, and stratified models of probabilistic processes, Inf. Comput. 121 (1995) 59–80.
[31] N. Götz, U. Herzog, M. Rettelbach, Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras, in: L. Donatiello, R. Nelson (Eds.), Performance Evaluation of Computer and Communication Systems, LNCS 729, Springer, Berlin, 1993, pp. 121–146.
[32] N. Götz, U. Herzog, M. Rettelbach, TIPP – Introduction and application to protocol performance analysis, in: H. König (Ed.), Formale Beschreibungstechniken für Verteilte Systeme, Saur publishers, 1993.
[33] E. Goubault, Durations for truly-concurrent transitions, in: H.R. Nielson (Ed.), Programming Languages and Systems – ESOP'96, LNCS 1058, Springer, Berlin, 1996, pp. 173–188.
[34] J. Gunawardena, A dynamic approach to timed behaviour, in: B. Jonsson, J. Parrow (Eds.), Concur' 94: Concurrency Theory, LNCS 836, Springer, Berlin, 1994, pp. 178–193.
[35] H. Hansson, Time and probability in formal design of distributed systems, Ph.D. Thesis, Uppsala University, 1991.
[36] H. Hansson, B. Jonsson, A calculus for communicating systems with time and probabilities, in: Proc. 11th IEEE Real-Time Systems Symp., IEEE Computer Press, 1990, pp. 278–287.
[37] P.G. Harrison, B. Strulo, Stochastic process algebra for discrete event simulation, in: F. Baccelli, A. Jean-Marie, I. Mitrani (Eds.), Quantitative Methods in Parallel Systems, Springer, Berlin, 1995, pp. 18–37.
[38] M. Hennessy, T. Regan, A temporal process algebra, Inf. Comput. 117 (1995) 221–239.
[39] H. Hermanns, M. Rettelbach, Syntax, semantics, equivalences, and axioms for MTIPP, in: U. Herzog, M. Rettelbach (Eds.), Proc. 2nd Int. Workshop on Process Algebra and Performance Modelling, Arbeitsbericht 27 (4), Universität Erlangen-Nürnberg, 1994, pp. 71–88.

[40] H. Hermanns, U. Herzog, V. Mertsiotakis, Stochastic process algebras: between LOTOS and Markov chains, Comput. Netw. ISDN Syst. 30 (1998) 901–924; this issue.
[41] U. Herzog, A concept for graph-based stochastic process algebras, generally distributed activity times, and hierarchical modelling, in: M. Ribaudo (Ed.), Proc. 4th Int. Workshop on Process Algebra and Performance Modelling, C.L.U.T. Press, 1996, pp. 1–20.
[42] J. Hillston, A compositional approach to performance modelling, Ph.D. Thesis, University of Edinburgh, 1994.
[43] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, 1985.
[44] ISO IS 8807, LOTOS – A formal description technique based on the temporal ordering of observational behaviour, 1989.
[45] W. Janssen, M. Poel, Q. Wu, J. Zwiers, Layering of real-time distributed processes, in: H. Langmaack, W.-P. de Roever and J. Vytopil (Eds.), Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863, Springer, Berlin, 1994, pp. 393–417.
[46] J-P. Katoen, Quantitative and qualitative extensions of event structures, Ph.D. Thesis, University of Twente, 1996.
[47] J-P. Katoen, E. Brinksma, D. Latella, R. Langerak, Stochastic simulation of event structures, in: M. Ribaudo (Ed.), Proc. 4th Int. Workshop on Process Algebra and Performance Modelling, C.L.U.T. Press, 1996, pp. 21–40.
[48] J-P. Katoen, R. Langerak, D. Latella, Modelling systems by probabilistic process algebra: an event structures approach, in: R.L. Tenney, P.D. Amer, M.Ü. Uyar (Eds.), Formal Description Techniques VI, North-Holland, Amsterdam, 1994, pp. 253–268.
[49] J-P. Katoen, R. Langerak, E. Brinksma, D. Latella, T. Bolognesi, A consistent causality-based view on a timed process algebra including urgent interactions, J. Formal Methods Syst. Design 12 (2) (1998) 189–216 (extended abstract in: A. Cornell, D. Ionescu (Eds.), Proc. 3rd Amast Workshop on Real-Time System Development, 1996, pp. 212–227).
[50] J-P. Katoen, D. Latella, R. Langerak, E. Brinksma, On specifying real-time systems in a causality-based setting, in: B. Jonsson, J. Parrow (Eds.), Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 1135, Springer, Berlin, 1996, pp. 385–405.
[51] R. Langerak, Transformations and semantics for LOTOS, Ph.D. Thesis, University of Twente, 1992.
[52] R. Langerak, Bundle event structures: a non-interleaving semantics for LOTOS, in: M. Diaz, R. Groz (Eds.), Formal Description Techniques V, North-Holland, Amsterdam, 1993, pp. 331–346.
[53] R. Langerak, E. Brinksma and J-P. Katoen, Causal ambiguity and partial orders in event structures, in: A. Mazurkiewicz, J. Winkowski (Eds.), Concur'97: Concurrency Theory, LNCS 1243, Springer, Berlin, 1997, pp. 317–332.
[54] L. Léonard, G. Leduc, An introduction to ET-LOTOS for the description of time-sensitive systems, Comput. Netw. ISDN Syst. 29 (3) (1997) 271–292.
[55] R. Loogen, U. Goltz, Modelling nondeterministic concurrent processes with event structures, Fund. Inf. 14 (1991) 39–74.
[56] G. Lowe, Representing nondeterminism and probabilistic behaviour in reactive processes, Technical Report PRG-TR-11-93, Oxford University, 1993.
[57] G. Lowe, Probabilistic and prioritized models of timed CSP, Th. Comp. Sci. 138 (1995) 315–352.
[58] A. Maggiolo-Schettini, J. Winkowski, Towards an algebra for timed behaviours, Th. Comp. Sci. 103 (1992) 335–363.
[59] F. Moller, C. Tofts, A temporal calculus of communicating systems, in: J.C.M. Baeten and J.-W. Klop (Eds.), Concur'90: Theories of Concurrency – Unification and Extension, LNCS 458, Springer, Berlin, 1990, pp. 401–415.
[60] C. Miguel, A. Fernández, L. Vidaller, LOTOS extended with probabilistic behaviours, Formal Aspects Comput. 5 (1993) 253–281.
[61] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, 1989.
[62] D. Murphy, Time and duration in noninterleaving concurrency, Fund. Inf. 19 (1993) 403–416.
[63] M.F. Neuts, Matrix-Geometric Solutions in Stochastic Models – An Algorithmic Approach, Johns Hopkins University Press, 1981.
[64] X. Nicollin, J. Sifakis, An overview and synthesis on timed process algebras, in: J.W. de Bakker, C. Huizing, W.-P. de Roever, G. Rozenberg (Eds.), Real-Time: Theory in Practice, LNCS 600, Springer, Berlin, 1992, pp. 526–548.
[65] M. Nielsen, G.D. Plotkin, G. Winskel, Petri nets, event structures and domains, Th. Comp. Sci. 13 (1) (1981) 85–108.
[66] M. Núñez, D. de Frutos, Testing semantics for probabilistic LOTOS, in: G. von Bochmann, R. Dssouli, O. Rafiq (Eds.), Formal Description Techniques VIII, Chapman and Hall, New York, 1995, pp. 365–380.
[67] G.M. Pinna, A. Poigné, On the nature of events: another perspective in concurrency, Th. Comp. Sci. 138 (2) (1995) 425–454.
[68] C. Priami, Stochastic π-calculus with general distributions, in: M. Ribaudo (Ed.), Proc. 4th Int. Workshop on Process Algebra and Performance Modelling, C.L.U.T. Press, 1996, pp. 41–57.
[69] S. Purushothaman, P.A. Subrahmanyam, Reasoning about probabilistic behaviour in concurrent systems, IEEE Trans. Softw. Eng. 13 (6) (1987) 740–745.
[70] J. Quemada, D. de Frutos, A. Azcorra, TIC: A TImed Calculus, Formal Aspects Comput. 5 (1993) 224–252.
[71] G.M. Reed, A.W. Roscoe, A timed model for Communicating Sequential Processes, Th. Comp. Sci. 58 (1988) 249–261.
[72] A. Rensink, Models and methods for action refinement, Ph.D. Thesis, University of Twente, 1993.
[73] N. Rico, G. von Bochmann, Performance description and analysis for distributed systems using a variant of LOTOS, in: B. Jonsson, J. Parrow, B. Pehrson (Eds.), Protocol Specification, Testing, and Verification IX, North-Holland, Amsterdam, 1991, pp. 199–213.
[74] R.A. Sahner, K.S. Trivedi, Performance and reliability analysis using directed acyclic graphs, IEEE Trans. Softw. Eng. 13 (10) (1987) 1105–1114.
[75] S. Schneider, An operational semantics for timed CSP, Inf. Comput. 116 (1995) 193–213.
[76] K. Seidel, Probabilistic communicating processes, Th. Comp. Sci. 152 (1995) 219–249.
[77] G.S. Shedler, Regenerative Stochastic Simulation, Academic Press, New York, 1993.

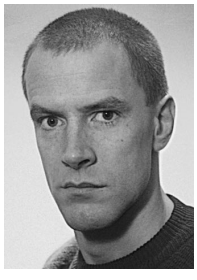
[78] R. Sisto, L. Ciminiera, A. Valenzano, Probabilistic characterization of algebraic protocol specifications, in: Proc. 12th Int. Conf. on Distributed Computing Systems, IEEE Comput. Press, 1992, pp. 260–268.
[79] C.M.N. Tofts, A synchronous calculus of relative frequency, in: J.C.M. Baeten and J.-W. Klop (Eds.), Concur'90: Theories of Concurrency – Unification and Extension, LNCS 458, Springer, Berlin, 1990, pp. 467–480.
[80] F.W. Vaandrager, A simple definition for parallel composition of prime event structures, Report CS-R8903, Centre for Mathematics and Computer Science, 1989.
[81] G. Winskel, An introduction to event structures, in: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (Eds.), Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354, Springer, Berlin, 1989, pp. 411–427.
[82] W.Yi, Real-time behaviour of asynchronous agents, in: J.C.M. Baeten and J.-W. Klop (Eds.), Concur'90: Theories of Concurrency – Unification and Extension, LNCS 458, Springer, Berlin, 1990, pp. 501–520.
[83] W. Yi, K.G. Larsen, Testing probabilistic and nondeterministic processes, in: R.J. Linn, M.Ü. Uyar (Eds.), Protocol Specification, Testing, and Verification XII, North-Holland, Amsterdam, 1992, pp. 47–61.
[84] J.J. Žic, Time-constrained buffer specifications in CSP + T and timed CSP, ACM Trans. Prog. Lang. and Syst. 16 (6) (1994) 1661–1674.



Ed Brinksma holds the chair of Formal Methods and Tools of the Faculty of Computer Science at the University of Twente at Enschede, The Netherlands. His main research interest lies in the application of formal methods to the design of distributed systems, including specification, verification, implementation, testing, and software tool support. Ed was chairman of the ISO standardisation committee of the formal specification language LOTOS (ISO IS 8807). His current research focusses on testing (test derivation), validation by model checking, formal methods for real-time systems, linking performance models to formal specifications, and the introduction of formal methods into industrial working practices.



Joost-Pieter Katoen received his M.Sc. degree (with honours) and Ph.D. degree in Computer Science from the University of Twente (NL) in 1987 and 1996, respectively. He is currently a research associate at the Faculty of Computer Science at the University of Erlangen-Nürnberg (D). From 1988–1990 he was a postgraduate student at the Eindhoven University of Technology (NL) and from 1990–1992 a research scientist at Philips Research Laboratories (NL). His research interest lies in formal methods and applications thereof in a broad sense. In particular: specification, design, and verification of distributed systems, semantic models of process algebras (and quantitative extensions thereof), and performance evaluation based on formal specifications. In addition, automatic code generation and the design of concurrent programs do interest him.



Rom Langerak graduated cum laude in Applied Mathematics at the University of Twente, The Netherlands, in 1986. He received his Ph.D. from the Department of Computer Science in 1992 and is currently an assistant professor at the same department. His research interest lies in the application of formal methods to the design of distributed systems, including specification, verification, implementation and testing.

Diego Latella received his Laurea ''cum Laude'' in Scienze dell'Informazione from the University of Pisa (I) in 1983. During 1984–1985 he was research associate with the Department of Computer Science of the University of Pisa. In 1986 he joint CNUCE, an Institute of the Italian National Research Council. He has been lecturing at the Department of Computer Science of the University of Pisa (1988-1995). In 1992 he was Academic Visitor of the University of Twente. His current main research interests fall in the area of specification and verification models and tools for concurrent systems. In particular he is interested in quantitative extensions of models for concurrency, mainly timed and/or probabilistic and/or stochastic process algebras. His specific current area of research is that of true concurrency (event structures) semantic models for the above process algebras. Moreover, he is working in modelling and validating real-life systems by means of automatic specification and validation tools. He is also interested in functional approaches to concurrency and Data-Flow model of computation for non-deterministic systems. In addition, he has some interest in the application of abstract interpretation to specification, analysis, transformation and to verification. He is member of the European Association for Theoretical Computer Science – EATCS, the ERCIM Working Group on Formal Methods for Industrial Critical Systems (Chairman), the International Network of Engineers and Scientists for Global Responsibility – INES (Council Member).