Available online at www.sciencedirect.com

Computer Standards & Interfaces 29 (2007) 205 – 215

COMPUTER STANDARDS & INTERFACES

www.elsevier.com/locate/csi

Enhanced event structures: Towards a true concurrency semantics for E-LOTOS

M. Kapus-Kolar *

Jožef Stefan Institute, Jamova 39, SI-1111 Ljubljana, Slovenia

Received 19 September 2005; accepted 25 March 2006
Available online 10 July 2006

Abstract

E-LOTOS is a standard process-algebraic language for formal specification of real-time concurrent and reactive systems. Its originally defined semantics is based on interleaving of events. In the present paper, we propose an enhanced kind of event structures and show how to employ them to give E-LOTOS processes a branching-time true concurrency semantics. The proposed event structures can model real-time processes with data handling and excel in concise representation of event renaming and synchronization.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Event structures; Process algebra; E-LOTOS; Formal semantics; True concurrency

1. Introduction

E-LOTOS [4,19], an enhanced successor of LOTOS [3,1], is one of the standard languages for formal specification of real-time concurrent and reactive systems. According to the operational semantics given in [4], an E-LOTOS specification characterizes a process by its readiness to engage into various kinds of atomic instantaneous events, where all internal process events are by definition anonymous and all concurrent events are represented as interleaved. That reflects the fact that LOTOS was originally intended for specification of "temporal ordering of observational behaviour" [3], where the possibility of simultaneous events was neglected.

Such characterization of a process often fails to provide sufficient information for its further refinement. For example, when refining an event into a process, one must know the relations of causality and conflict in which the event is engaged, because at least some of the events constituting its refinement are supposed to inherit them [6]. One might also want to identify events which are truly concurrent, so that their execution may be delegated to different concurrent components of a system. Hence, it is convenient to model a process by its events and their relationships, i.e., by its *event structure*.

With their detailed representation of process behaviour, event-structure models are ideal not only for refinement of events, but also for refinement of their relationships, necessary, for example, when designing a distributed implementation of a process, i.e., refining the relationships into a coordination protocol [16]. On the other hand, event structures refrain from modelling process architecture, following the idea that grouping of events into subprocesses is, like their assignment to gates, just a matter of interpretation [12]. Hence, an event structure is indeed just a collection of events and relationships, i.e., a set of orthogonal process properties.

The syntactic and semantic simplicity makes event structures easy to use and ideal for incremental design. Elements of an event structure may be added and removed at will, although it is advisable to take care that the structure stays within a class which can be easily manipulated with the available tools (e.g., to avoid causal ambiguity [20]). As increasingly more powerful manipulation tools are available, we in this paper limit our attention to the expressiveness of event structures.

A process-algebraic specification describes a set of elementary processes and their hierarchical composition. Elementary processes often correspond to individual events and composition operators provide information on their relationships. The problem is that when the operands of a composition operator are themselves compound processes, the relationships between their constituent events are described only implicitly. To

* Tel.: +386 14773531; fax: +386 14262102.
E-mail address: monika.kapus-kolar@ijs.si.

0920-5489/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2006.03.007

overcome the problem, process-algebraic languages are being furnished with event-structure semantics.

For LOTOS without data, an event-structure semantics was proposed in [13]. The semantics was extended to timed processes [11], and subsequently employed [2] (still without data) for ET-LOTOS [14], a predecessor of E-LOTOS. In the present paper, we propose an event-structure semantics for E-LOTOS.

The proposed true concurrency semantics for E-LOTOS is not the only contribution of the paper. Perhaps even more important is the newly developed kind of event structures, which we name *enhanced event structures*, because in the name E-LOTOS, "E" stands for "enhanced." Enhanced event structures can model real-time processes with data handling and excel in concise representation of event renaming and synchronization.

The paper is organized as follows: In Section 2, we analyse the intuitive semantics of E-LOTOS processes and gradually develop a kind of event structures sufficiently expressive for their elegant modelling. Section 3 contains a detailed study of event-structure semantics of elementary E-LOTOS processes and of individual process composition operators. Section 4 comprises a discussion and conclusions.

2. Enhanced event structures

2.1. Events

The main objects in an event structure $\mathcal{E}$ are its events e, collected in an E. An e represents atomic, instantaneous execution of some tasks. Hence, it can be seen also as a Boolean variable jumping from false (e has not yet occurred) to true (e has already occurred).

The elementary events of an E-LOTOS process B, i.e., the potential events of its elementary subprocesses, with no doubt, correspond to the event concept defined above, and as such qualify for inclusion into E of the event structure modelling B. The past practice has been to also include into E the compound events of B [13,11,2], i.e., interactions of its subprocesses. We find the approach contra-intuitive, because in the original E-LOTOS semantics, the behaviour of a B in no way depends on how its already executed elementary events have synchronized. In other words, while a B remembers the occurrence of its elementary events, it does not remember the occurrence of its compound events. Therefore, we propose that the members of E are exactly the potential elementary events of the modelled B. Thereby, the size of E is kept proportional to the number of the elementary subprocesses of B, while otherwise it could grow exponentially with the number of the parallel compositions specified for the processes.

2.2. Preconditions and their triggers

An e can occur only if it is currently logically enabled, i.e., if it has not been disabled or if all its disablings have been cancelled. In E-LOTOS, an e in a B might be disabled from the beginning or upon the occurrence of a disabling e', while cancellation of a disabling, i.e., enabling or re-enabling of e, requires appropriate values of the input parameters of B and/or execution of a set of events whose occurrence and the data they have generated justify the cancellation.

In LOTOS or ET-LOTOS, an already enabled e might be disabled, but is never re-enabled. Hence, enabling and disabling of events, i.e., the relations of causality and conflict, are two separate issues. In E-LOTOS, its suspend/resume operator introduces resolvable conflicts. Examples of event structures capable to model resolvable conflicts are [5,15,7], but none of them addresses data and time.

Trying to represent enabling and disabling of events in an integrated manner, we observe that for every E-LOTOS event e, there exists a (possibly empty) set of preconditions, constraints expected to be satisfied just before the occurrence of e. In an $\mathcal{E}$, let preconditions ξ be objects of a special kind, collected in a Ξ. For an e, let $\Xi(e)$ list the associated ξ. For a ξ, let $E(\xi)$ list the e with ξ in $\Xi(e)$.

For every ξ and e in $E(\xi)$, we introduce a Boolean trigger $\tau(e,\xi)$ and define that e is logically enabled if for every ξ in $\Xi(e)$, $\tau(e,\xi)$ is false or ξ is true. Besides, if a $\tau(e,\xi)$ is false just before the occurrence of e, it must be false also just after it. Hence, a $\tau(e,\xi)$ indicates a disabling of e for which ξ is the cancelling condition. The approach is a generalization of that in [5].

2.3. Postconditions and their triggers

In E-LOTOS, the exact manner in which a gate event will occur (what its execution time and other generated data will be) is not known in advance, but there is in principle a constraint expected to be satisfied just after the occurrence, i.e., a postcondition constraint. In an $\mathcal{E}$, let postconditions λ be objects of a special kind, collected in a Λ. For an e, let $\Lambda(e)$ list the associated λ. For a λ, let $E(\lambda)$ list the e with λ in $\Lambda(e)$.

For every λ and e in $E(\lambda)$, we introduce a Boolean trigger $\tau(e,\lambda)$ and define that just after the occurrence of e, every λ in $\Lambda(e)$ with $\tau(e,\lambda)$ true just before e must be true. Selectively triggered postconditions facilitate full control of event simultaneity, like the event structures proposed in [15]. One might want to use them to selectively restrict simultaneity of otherwise independent events.

Example 1. To specify that an e_1 must occur simultaneously with an e_2 or an e_3, but not with both, one could introduce into $\Lambda(e_1)$ postconditions "$e_2 \wedge e_3$", "$e_2 \oplus e_3$" and "false", with triggers "$e_2 \oplus e_3$", "$\neg e_2 \wedge \neg e_3$" and "$e_2 \wedge e_3$", respectively.

2.4. Aging and urgency

In the following, the term "time" denotes the internal time of an evolving $\mathcal{E}$ or the modelled process, i.e., the value of its clock, that starts from 0. For simplicity, we assume that the time domain is discrete. During the dynamic evolution of an $\mathcal{E}$, the flow of time is reflected in the aging of its events. Whenever the clock increases, so does the age $a(e)$ of every e currently in the state of idling. An e is in the state of idling when it has not yet occurred, but is logically enabled. When an e finally occurs, its