Two Approaches to Event Definition

Antony Galton[1] and Juan Carlos Augusto[2]

[1] School of Engineering and Computer Science, University of Exeter, UK
A.P.Galton@exeter.ac.uk
[2] Department of Electronics and Computer Science, University of Southampton, UK
jca@ecs.soton.ac.uk

Abstract. We compare two approaches to event definition, deriving from the Active Database and Knowledge Representation communities. We relate these approaches by taking a system of the former kind, displaying some of its shortcomings, and rectifying them by remodelling the system in the latter style. We further show the extent to which the original system can be recreated within the remodelled system. This bridge between the two approaches should provide a starting point for fruitful interaction between the two communities.

1 Events in Knowledge Representation and Databases

Events are ubiquitous in both life and computing. In both areas we frequently classify events, considering many individual events together as instances of some class, or event-type, to which we give a name (for example "plane crashes" or "user interrupts"). Each event-type gives rise to its own particular set of problems, but we may identify two as fundamental: 'How is the event-type defined?' and 'How can we detect when an event of that type has occurred?'. These two questions are clearly closely interrelated, but they are emphatically not the same question.

There are many areas of computing where event definition and detection are important. In this paper we focus on two areas representing two different approaches to events, which we have found it instructive to compare. These are knowledge representation and active databases. The former is generally more familiar to the AI community, the latter to the database community, but from our investigations it is clear that the concerns of both areas are relevant to both communities. Despite this, work in the two areas has for the most part proceeded separately, leading to divergence of notation and terminology. We believe it will benefit both areas to attempt to bridge this gap.

A database in the traditional, 'passive' sense is merely a repository (albeit perhaps highly structured) of information, which provides answers to queries posed by the user. The 'events' in the life of such a database are essentially the user-initiated updates and queries, and the responses to both types of events follow a rigid preprogrammed pattern. In particular, the database system itself is unable to produce behaviour except in direct response to individual external triggers. By contrast, an *active* database monitors the succession of events impinging on it from outside, and is endowed with the ability to detect not only events but also significant *patterns* of events, and to respond to these in accordance with some set of 'event-condition-action' (ECA) rules. Thus an active

database endowed with a set of ECA rules is in a certain sense capable of independent action, only indirectly triggered by inputs from its users.

A key feature of any active database system is its capacity to detect composite events of various descriptions. Systems may differ with respect to the available classes of primitive events, the variety of operators used for defining composite events from primitive ones, and the mechanisms by which composite events are detected. For example, the COMPOSE system of [1] provides four classes of primitive event (*object state events*, *method execution events*, *time events*, and *transaction events*), and a set of event-composition operators. The operators are selected so that their expressive power is exactly the same as that of regular grammars; this enables the event-detection mechanism to be implemented using finite automata, each event being associated with an automaton which reaches an accepting state exactly when that event occurs.

Another system, SAMOS [2, 3], combines active and object-oriented features in a single framework. Primitive events are *time events*, *method events*, *transaction events*, and *abstract events*, while composite events are built up from these using operators such as disjunction, conjunction, and sequential composition. Associated with each kind of primitive event is a method by which events of that kind are detected. These methods are used as the basis for a coloured Petri net which enables the detection of composite events. For each operator of the event algebra there is an associated structure in the net; these structures can be combined recursively as needed. When the primitive-event detector is activated the corresponding node in the net is activated and certain transitions fire. If as a result of these firings the node associated with some composite event acquires a token, then the composite event has been detected. The behaviour of the net is implemented by means of matrix-processing algorithms.

SNOOP [4] is another system emanating from the database community. We defer a description of this system to section 3, as it will provide the starting-point for the investigations reported in this paper.

Finally, we mention the work of [5], which provides a language for expressing event-based conditions in an OODB framework. As usual, basic and composite events are considered, the latter obtained by means of a predefined set of event constructors applied to other basic or composite events. Database events can be classified as *instantaneous* or *persistent*, while temporal events can be *absolute* (e.g., the first hour of a specific day), *periodic* (e.g., each working day), or *relative* (e.g., five hours after a modification to the database). The language resembles that of SNOOP, although the authors claim that it is strictly more expressive.

A feature that is common to all these systems is that all events, whether primitive or composite, are considered to be instantaneous, that is, the time associated with the event is an instant rather than an interval. In the case of a composite event built up from events occurring at different times, the associated instant is usually the time of the *last* of its contributory primitive events. This constraint is natural enough in a context where the prime focus of interest is on event detection, since typically a composite event will be detected at the time that its last contributory component is detected. However, this does lead to logical difficulties in the case of some composite event-types, as will be explained below. The only work on active databases we are aware of in which events are treated as having positive duration is [6]

Turning now to knowledge representation, one of the best-known proposals for the formalisation of temporal reasoning is the Interval Calculus [7, 8]. Here *all* events are regarded as durative, that is taking time, and accordingly intervals (rather than instants) are considered to be the basic temporal concept. A set of 13 basic relations between intervals is defined, and the rules governing the composition of such relations is regarded as a key factor in controlling temporal reasoning. Allen distinguishes between *properties*, *processes*, and *events*, the latter being related to intervals by means of a predicate $Occurs(E, i)$ which is used to assert that an event of type E occurs on the interval i. Events are assumed to be unitary in the sense of the formula $Occurs(E, i) \wedge In(i', i) \rightarrow \neg Occurs(E, i')$, where $In(i', i)$ means that i' is a *proper* subinterval of i. The main focus of this work is on the development of axioms to formalise the notions of action and causality. Event types are characterised in terms of necessary and sufficient conditions for their occurrence.

Event definition in terms of occurrence conditions also plays a prominent part in the Event Calculus of [9] (which is connected with temporal deductive databases rather than active databases), and in the work of Galton [10]. Rather than detection of events as they occur, the knowledge representation work on events has mainly concentrated on what inferences can be made from the fact that certain events are known to have occurred. These inferences may extend to the possibility of future events occurring, thus providing a link to another important theme of AI, planning. An exception is the work of [11], whose IxTeT system detects the occurrence of *chronicles* specified as conjunctions of instantaneous events together with constraints on their relative times of occurrence.

In the remainder of the paper, we first set up a simple discrete temporal framework (this is for convenience; the same issues arise in continuous time), then examine the SNOOP system in some detail as representative of the active database systems in which events are regarded as occurring at their time of detection and hence as instantaneous. In effect, events are defined in terms of their *detection conditions*. We next illustrate the problems that arise from the decision to treat events in this way, and go on to show that the problems do not arise if we adopt instead the assumption widespread in the knowledge representation community that events should be treated as durative and defined in terms of their *occurrence conditions*. We then investigate how far it is possible to reconcile the two accounts of events; we associate each event with a 'detection event' in such a way that detection of an event, in the database sense, can be exactly identified with the occurrence, in the knowledge representation sense, of its associated 'detection event'. The key question is whether the occurrence conditions of the latter can be precisely given; by determining precisely when they can and when they cannot, we provide a measure of the extent of the disagreement between the two approaches. Although this is only a beginning, we believe that this work can provide the foundation for a productive dialogue between the active database and knowledge representation communities.

2 A Temporal Framework

Whereas time is commonly conceptualized as a continuum, it is sufficient for many applications to treat it as a discrete system modelled by the integers under their usual

ordering. In such a model, individual integers represent *atomic intervals*, chunks of time that are regarded as indivisible. It is assumed that no change can take place within an atomic interval. Longer intervals are specified by their initial and final atomic intervals; thus the interval [4,7] is composed of the atomic intervals 4, 5, 6, and 7. It is sometimes convenient to allow the notation [4,4] to refer to the atomic interval 4.

The main focus of interest in this paper will be on *events*. We distinguish the occurrence of an event from its detection. Typically an event is, or can be, detected at the end of the interval over which it occurs. We shall assume that event detection always occurs at an atomic interval, whereas the event itself may occur over an extended interval. We shall use the notation $\mathbf{D}(E, t)$ to mean that an event of type E is detected at time t, and $\mathbf{O}(E, [t, t'])$ to mean that such an event occurs over the interval $[t, t']$. An event-type which can only occur on an atomic interval will be called an *atomic event*. Events may be characterized either in terms of their detection conditions, or in terms of their occurrence conditions. These two approaches lead to rather different conceptions of events, which it will be one purpose of this paper to reconcile.

We assume that we are provided with a stock of *primitive event types*; we say nothing about how these are specified (although the primitive event types recognised by SNOOP are briefly described below), but shall assume that for each primitive event type E all facts of the form $\mathbf{O}(E, [t, t'])$ and $\mathbf{D}(E, t)$ are known. Our main interest in this paper will be in *composite event types*, which are constructed from the primitive event types by means of a set of operators. In the literature many such operators have been considered. In this paper we shall discuss those of the system SNOOP introduced by Chakravarty et al. [4].

3 Detectable Event-types

An event-type defined in terms of its detection condition will be called a *detectable* event-type. To illustrate the use of detectable event types we describe the SNOOP system of [4]. In this system events are either primitive or composite, the latter being constructed from the former using the operators defined in the paper. Amongst primitive events are distinguished

1. database events, corresponding to database operations such as $retrieve$, $insert$, $update$, and $delete$;
2. temporal events, which pick elements of the passage of time itself, either absolutely (e.g., calendar dates, clock times) or relatively (in terms of some reference event, e.g., 3 seconds after a database update);
3. explicit events, which include any events detected by other application programs and input as primitive events into the DBMS.

The reader is referred to [4] for the original definitions of the operators used to form composite events. Here we attempt to provide equivalent definitions within the formalism laid out in the previous section; in cases where we found the original definitions ambiguous or unclear, we have resorted to informed guesswork in formulating the exact equivalents in our system. We consider the operators in the order they appear in the original paper.

or Disjunction of events: $\mathbf{D}(E_1 \triangledown E_2, t) \stackrel{\text{def}}{=} \mathbf{D}(E_1, t) \vee \mathbf{D}(E_2, t)$.
and Conjunction of events. The events are not required to be simultaneous.

$$\mathbf{D}(E_1 \triangle E_2, t) \stackrel{\text{def}}{=} \exists t' \leq t((\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_2, t)) \vee (\mathbf{D}(E_1, t) \wedge \mathbf{D}(E_2, t'))$$

any The original definition of this operator used a second-order condition, with a variable number of quantifiers depending on the number of events considered; in order to work within first-order logic, we define a series of "any" operators. For $m \leq n$, the event $\mathsf{ANY}_n^m(E_1, \ldots, E_n)$ occurs when m events out of n distinct specified events occur, in any order:

$$\begin{aligned}\mathbf{D}(\mathsf{ANY}_n^m(E_1, \ldots, E_n), t) \stackrel{\text{def}}{=} \exists t_1 \leq \ldots \leq t_m = t\, \exists i_1, \ldots, i_m \in \{1, \ldots, n\} \\ (\#\{i_1, \ldots, i_m\} = m \wedge \mathbf{D}(E_{i_1}, t_1) \wedge \cdots \wedge \mathbf{D}(E_{i_m}, t_m)).\end{aligned}$$

The first conjunct on the right-hand side ensures that the E_{i_j} are all distinct (here $\#X$ is the cardinality of X).
seq The sequential composition of two events:

$$\mathbf{D}(E_1; E_2, t) \stackrel{\text{def}}{=} \exists t' < t(\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_2, t)).$$

A According to Chakravarthy et al., the 'aperiodic' operator A allows one to express the occurrence of an event E_2 within the interval defined by two other events E_1 and E_3. Their definition seems to express something different, however, namely that E_2 occurs within an interval beginning with E_1 at a time when E_3 has not yet occurred. There is no requirement for the event E_3 to occur at all. We thus have the definition:

$$\begin{aligned}&\mathbf{D}(\mathsf{A}(E_1, E_2, E_3), t) \stackrel{\text{def}}{=} \\ &\quad \mathbf{D}(E_2, t) \wedge \exists t' < t(\mathbf{D}(E_1, t') \wedge \forall t''(t' \leq t'' \leq t \rightarrow \neg\mathbf{D}(E_3, t'')))\end{aligned}$$

This can be expressed more simply by the introduction of a new predicate $\mathbf{D}_{in}$ which says that an event is detected at *some* point within a stated interval:

$$\mathbf{D}_{in}(E, [t_1, t_2]) \stackrel{\text{def}}{=} \exists t(t_1 \leq t \leq t_2 \wedge \mathbf{D}(E, t)).$$

We can then put

$$\mathbf{D}(\mathsf{A}(E_1, E_2, E_3), t) \stackrel{\text{def}}{=} \mathbf{D}(E_2, t) \wedge \exists t' < t(\mathbf{D}(E_1, t') \wedge \neg\mathbf{D}_{in}(E_3, [t', t]))$$

P The 'periodic' operator P caused us even more difficulty than A. However, we believe that $\mathsf{P}(E_1, n, E_3)$ is an event type which occurs every n time-steps after an occurrence of E_1 so long as E_3 does not occur. Note that, contrary to what one might think at first, the periodic operator does not express the periodic recurrence of some detectable event, but only that a certain period of time has elapsed a whole number of times since a given detectable event. It is thus a 'virtual' event rather than a 'real' one.

$$\begin{aligned}\mathbf{D}(\mathsf{P}(E_1, n, E_3), t) \stackrel{\text{def}}{=} \exists t' < t \exists i \in \mathbb{Z}^+(t = t' + ni \wedge \mathbf{D}(E_1, t') \wedge \\ \neg\mathbf{D}_{in}(E_3, [t' + 1, t]))\end{aligned}$$

not The non-occurrence of the event E_2 in an interval delimited by occurrences of E_1 and E_3:

$$\mathbf{D}(\neg(E_2)[E_1, E_3], t) \stackrel{\text{def}}{=} \exists t' < t(\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_3, t) \wedge \neg\mathbf{D}_{in}(E_2, [t', t]))$$

4 Why Detection Conditions are Inadequate

All the event-types defined in the previous section are detectable, assuming that all the event-types E_1, E_2, E_3 occurring in the definitions are detectable. This is because the events are defined in terms of detection conditions and not in terms of occurrence conditions. However, this leads to some problems; it means that the events that are defined are not always exactly the events that are, presumably, intended.

This is revealed when we consider a composite event such as $E_1; (E_2; E_3)$. Intuitively, since ';' expresses sequential composition, we should expect this event to occur whenever an occurrence of E_1 is followed by an occurrence of E_2 which is in turn followed by an occurrence of E_3. It should therefore be a different event from $E_2; (E_1; E_3)$. Now consider the detection conditions for the two events:

$$\begin{aligned}
\mathbf{D}(E_1; (E_2; E_3), t) &\iff \exists t' < t(\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_2; E_3, t)) \\
&\iff \exists t' < t(\mathbf{D}(E_1, t') \wedge \exists t'' < t(\mathbf{D}(E_2, t'') \wedge \mathbf{D}(E_3, t))) \\
&\iff \exists t' < t, t'' < t(\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_2, t'') \wedge \mathbf{D}(E_3, t))
\end{aligned}$$

$$\begin{aligned}
\mathbf{D}(E_2; (E_1; E_3), t) &\iff \exists t' < t(\mathbf{D}(E_2, t') \wedge \mathbf{D}(E_1; E_3, t)) \\
&\iff \exists t' < t(\mathbf{D}(E_2, t') \wedge \exists t'' < t(\mathbf{D}(E_1, t'') \wedge \mathbf{D}(E_3, t))) \\
&\iff \exists t' < t, t'' < t(\mathbf{D}(E_2, t') \wedge \mathbf{D}(E_1, t'') \wedge \mathbf{D}(E_3, t))
\end{aligned}$$

The two detection conditions are equivalent, which means that as detectable events defined according to SNOOP, $E_1; (E_2; E_3)$ and $E_2; (E_1; E_3)$ are equivalent. This in turn means that ';' so defined does not, after all, express sequential composition but something different.

For another example, consider the event $\neg(E_2 \triangle E_2')[E_1, E_3]$, the negation of a conjunction. Intuitively, one might expect this to occur so long as at least one of E_2 and E_2' fails to occur in the interval delimited by E_1 and E_3. Consider, however, the detection conditions:

$$\begin{aligned}
&\mathbf{D}(\neg(E_2 \triangle E_2')[E_1, E_3], t) \\
&\quad\iff \exists t' \leq t(\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_3, t) \wedge \forall t''(t' \leq t'' \leq t \rightarrow \neg\mathbf{D}(E_2 \triangle E_2', t''))) \\
&\quad\iff \exists t' \leq t(\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_3, t) \wedge \forall t''(t' \leq t'' \leq t \rightarrow \\
&\qquad \neg\exists t^* \leq t''((\mathbf{D}(E_2, t^*) \wedge \mathbf{D}(E_2', t'')) \vee (\mathbf{D}(E_2, t'') \wedge \mathbf{D}(E_2', t^*))))) \\
&\quad\iff \exists t' \leq t(\mathbf{D}(E_1, t') \wedge \mathbf{D}(E_3, t) \wedge \forall t''(t' \leq t'' \leq t \rightarrow \\
&\qquad \forall t^* \leq t''\neg((\mathbf{D}(E_2, t^*) \wedge \mathbf{D}(E_2', t'')) \vee (\mathbf{D}(E_2, t'') \wedge \mathbf{D}(E_2', t^*)))))
\end{aligned}$$

We need not spell this out any further to see that it imposes a very strong condition on the detection of events of types E_2 and E_2': a condition which refers to *all* times t^* earlier than some time t'' in the interval of interest. This takes us right outside the interval to a consideration of the entire past history, clearly not in keeping with the intuitive meaning.

5 Event-types Defined in Terms of Occurrence

The problems encountered in the previous section all arise from the fact that the time associated with an event by the $Detect$ predicate is an atomic interval; the event itself

may occur over an extended interval, but in this case the $Detect$ predicate carries no information about how far into the past that interval extends. To remedy this, we shall redefine the SNOOP operators in terms of occurrence conditions rather than detection conditions; this is more in keeping with the approaches to events in the Knowledge Representation tradition, as described above. We shall refer to the system of redefined operators as O-SNOOP ('Occurrence-based SNOOP') as distinct from the original system which we shall refer to as D-SNOOP ('Detection-based SNOOP').

or $\mathbf{O}(E_1 \triangledown E_2, [t_1, t_2]) \stackrel{\text{def}}{=} \mathbf{O}(E_1, [t_1, t_2]) \vee \mathbf{O}(E_2, [t_1, t_2])$.

and $\mathbf{O}(E_1 \triangle E_2, [t_1, t_2]) \stackrel{\text{def}}{=} \exists t, t' (t_1 \leq t \leq t_2 \wedge t_1 \leq t' \leq t_2 \wedge$
$((\mathbf{O}(E_1, [t_1, t]) \wedge \mathbf{O}(E_2, [t', t_2])) \vee (\mathbf{O}(E_1, [t', t_2]) \wedge \mathbf{O}(E_2, [t_1, t])) \vee$
$(\mathbf{O}(E_1, [t_1, t_2]) \wedge \mathbf{O}(E_2, [t, t'])) \vee (\mathbf{O}(E_1, [t, t']) \wedge \mathbf{O}(E_2, [t_1, t_2])))$

any $\mathbf{O}(\mathsf{ANY}_n^m(E_1, \ldots, E_n), [t_1, t'_m]) \stackrel{\text{def}}{=}$
$\exists t_2, \ldots, t_m, t'_1, \ldots, t'_{m-1} \in [t_1, t'_m] \exists i_1, \ldots, i_m \in \{1, \ldots, n\}$
$(\#\{i_1, \ldots, i_m\} = m \wedge \mathbf{O}(E_{i_1}, [t_1, t'_{i_1}]) \wedge \cdots \wedge \mathbf{O}(E_{i_m}, [t_m, t'_{i_m}]))$.

seq We assume the events do not overlap:

$$\mathbf{O}(E_1; E_2, [t_1, t_2]) \stackrel{\text{def}}{=} \exists t, t' (t_1 \leq t < t' \leq t_2 \wedge \mathbf{O}(E_1, [t_1, t]) \wedge \mathbf{O}(E_2, [t', t_2]))$$

A The occurrence time for $\mathsf{A}(E_1, E_2, E_3)$ is the occurrence time for E_2—an occurrence of the event is an occurrence of E_2 in a certain context determined by E_1 and E_3. The rest of the occurrence condition specifies the context. There must be no occurrence of E_3 wholly within the interval between the occurrences of E_1 and E_2. To enable us to express this more concisely, we introduce a predicate $\mathbf{O}_{in}$ defined as follows:

$$\mathbf{O}_{in}(E, [t_1, t_2]) \stackrel{\text{def}}{=} \exists t'_1, t'_2 (t_1 \leq t'_1 \leq t'_2 \leq t_2 \wedge \mathbf{O}(E, [t'_1, t'_2])).$$

It will also be useful to define the end of an event by the rule:

$$\mathbf{O}(E\downarrow, t) \stackrel{\text{def}}{=} \exists t' \leq t \; \mathbf{O}(E, [t', t]).$$

We now have

$$\mathbf{O}(\mathsf{A}(E_1, E_2, E_3), [t_1, t_2]) \stackrel{\text{def}}{=} \mathbf{O}(E_2, [t_1, t_2]) \wedge \exists t < t_1 (\mathbf{O}(E_1\downarrow, t) \wedge \neg \mathbf{O}_{in}(E_3, [t, t_2])).$$

P As noted above, this operator expresses a virtual event which occurs at the moment of its detection; the occurrence time is therefore the same as the detection time. The non-occurrence of E_3 is handled as in the aperiodic case.

$$\mathbf{O}(\mathsf{P}(E_1, n, E_3), t) \stackrel{\text{def}}{=} \exists t' < t \exists i \in \mathbb{Z}^+ (t = t' + ni \wedge \mathbf{O}(E_1\downarrow, t') \wedge \neg \mathbf{O}_{in}(E_3, [t' + 1, t]))$$

not What is the time of a non-occurrence? Since it is non-occurrence of E_2 in a predefined interval, the only credible time to assign to it is just that interval. The interval extends from immediately after E_1 finishes to immediately before E_2 starts. To help us express the occurrence condition, we define the start of an event by the rule

$$\mathbf{O}(\uparrow E, t) \stackrel{\text{def}}{=} \exists t' (t \leq t' \wedge \mathbf{O}(E, [t, t'])).$$

We can now put

$$\mathbf{O}(\neg(E_2)[E_1, E_3], [t_1, t_2]) \stackrel{\text{def}}{=} \mathbf{O}(E_1{\downarrow}, t_1) \wedge \mathbf{O}(\uparrow E_3, t_2) \wedge \neg\mathbf{O}_{in}(E_2, [t_1, t_2]).$$

The problems we encountered with the D-SNOOP operators do not arise for O-SNOOP:

$$\begin{array}{l}
\mathbf{O}(E_1; (E_2; E_3), [t_1, t_2]) \\
\quad \Longleftrightarrow \ \exists t, t'(t_1 \leq t < t' \leq t_2 \wedge \mathbf{O}(E_1, [t_1, t]) \ \wedge \mathbf{O}(E_2; E_3, [t', t_2]) \\
\quad \Longleftrightarrow \ \exists t, t'(t_1 \leq t < t' \leq t_2 \wedge \mathbf{O}(E_1, [t_1, t]) \wedge \\
\qquad \exists t^*, t^{**}(t' \leq t^* < t^{**} \leq t_2 \wedge \mathbf{O}(E_2, [t', t^*]) \wedge \mathbf{O}(E_3, [t^{**}, t_2]))) \\
\quad \Longleftrightarrow \ \exists t, t', t^*, t^{**}(t_1 \leq t < t' \leq t^* < t^{**} \leq t_2 \wedge \\
\qquad \mathbf{O}(E_1, [t_1, t]) \wedge \mathbf{O}(E_2, [t', t^*]) \wedge \mathbf{O}(E_3, [t^{**}, t_2]))
\end{array}$$

$$\begin{array}{l}
\mathbf{O}(E_2; (E_1; E_3), [t_1, t_2]) \\
\quad \Longleftrightarrow \ \exists t, t'(t_1 \leq t < t' \leq t_2 \wedge \mathbf{O}(E_2, [t_1, t]) \wedge \mathbf{O}(E_1; E_3, [t', t_2]) \\
\quad \Longleftrightarrow \ \exists t, t'(t_1 \leq t < t' \leq t_2 \wedge \mathbf{O}(E_2, [t_1, t]) \wedge \\
\qquad \exists t^*, t^{**}(t' \leq t^* < t^{**} \leq t_2 \wedge \mathbf{O}(E_1, [t', t^*]) \wedge \mathbf{O}(E_3, [t^{**}, t_2]))) \\
\quad \Longleftrightarrow \ \exists t, t', t^*, t^{**}(t_1 \leq t < t' \leq t^* < t^{**} \leq t_2 \wedge \\
\qquad \mathbf{O}(E_2, [t_1, t]) \wedge \mathbf{O}(E_1, [t', t^*]) \wedge \mathbf{O}(E_3, [t^{**}, t_2]))
\end{array}$$

These two occurrence conditions are obviously inequivalent, and equally clearly in accordance with our intuitive understanding of triple sequential compositions.

In preparation for the next case, note that

$$\mathbf{O}_{in}(E_1 \triangle E_2, [t_1, t_2]) \iff \mathbf{O}_{in}(E_1, [t_1, t_2]) \wedge \mathbf{O}_{in}(E_2, [t_1, t_2]).$$

The proof is straightforward but rather tedious. We now have

$$\begin{array}{l}
\mathbf{O}(\neg(E_2 \triangle E_2')[E_1, E_3], [t_1, t_2]) \\
\qquad \Longleftrightarrow \ \mathbf{O}(E_1{\downarrow}, t_1) \wedge \mathbf{O}(\uparrow E_3, t_2) \wedge \neg\mathbf{O}_{in}(E_2 \triangle E_2', [t_1, t_2]) \\
\qquad \Longleftrightarrow \ \mathbf{O}(E_1{\downarrow}, t_1) \wedge \mathbf{O}(\uparrow E_3, t_2) \ \wedge \neg(\mathbf{O}_{in}(E_2, [t_1, t_2]) \wedge \mathbf{O}_{in}(E_2', [t_1, t_2])) \\
\qquad \Longleftrightarrow \ \mathbf{O}(E_1{\downarrow}, t_1) \wedge \mathbf{O}(\uparrow E_3, t_2) \ \wedge (\neg\mathbf{O}_{in}(E_2, [t_1, t_2]) \vee \neg\mathbf{O}_{in}(E_2', [t_1, t_2]))
\end{array}$$

which unlike in D-SNOOP does not involve reference to times indefinitely far into the past, and is clearly in accord with our intuitive understanding of the logic of this case.

6 D-SNOOP and O-SNOOP Compared

The time associated with any event by D-SNOOP is the time at which the event is detected; in almost every case this is the time of its last constituent, so that event E is detected at the time of its termination $E{\downarrow}$. We could regard $E{\downarrow}$ as the *detection event* of E, making the detection of any event equivalent to the occurrence of its detection event. D-SNOOP rests on the premiss that reasoning about composite events can be satisfactorily accomplished using detection events only. In order for this to work, it would be necessary for every event-composition operator Op to obey a rule of the form

$$Op(E_1, \ldots, E_n){\downarrow} \leftrightarrow Op(E_1{\downarrow}, \ldots, E_n{\downarrow}){\downarrow}$$

(Note that the final $\downarrow$ operator can be omitted in those cases when the composite is already instantaneous.) The D-SNOOP operators automatically obey this rule, but as we have seen, their interpretation is problematic. An exact measure of the extent to which our remodelled O-SNOOP operators agree with those of D-SNOOP is furnished by what proportion of them obey the above rule.

For four of the operators it can be straightforwardly verified that the rule does indeed hold; the following are all theorems of O-SNOOP:

$$\begin{aligned}(E_1 \triangledown E_2)\downarrow &= E_1\downarrow \triangledown E_2\downarrow \\ (E_1 \triangle E_2)\downarrow &= (E_1\downarrow \triangle E_2\downarrow)\downarrow \\ \mathsf{ANY}^m_n(E_1,\ldots,E_n)\downarrow &= \mathsf{ANY}^m_n(E_1\downarrow,\ldots,E_n\downarrow)\downarrow \\ \mathsf{P}(E_1,n,E_3)\downarrow &= \mathsf{P}(E_1\downarrow,n,E_3\downarrow)\end{aligned}$$

That part of D-SNOOP which handles these operators can thus be exactly recreated within O-SNOOP.

This leaves three problematic cases, namely sequential composition, the aperiodic operator, and negation—two of these are, not surprisingly, the operators we had trouble with earlier. In all these cases, the fact that in O-SNOOP events are, in general, durative, whereas detection events are always instantaneous, vitiates any attempt to express detection of the composite event in terms of detection of its components. For example, it is not possible to define $(E_1; E_2)\downarrow$ in terms of $E_1\downarrow$ and $E_2\downarrow$. The reason for this is that in order for $E_1; E_2$ to occur, it is necessary for E_1 to finish—and thus to be detectable—before E_2 starts. But the start of E_2 cannot be expressed in terms of the detection of E_2, which only refers to its end. Thus the closest we can come to such expressions is

$$\begin{aligned}(E_1; E_2)\downarrow &= (E_1\downarrow; E_2)\downarrow \\ A(E_1,E_2,E_3)\downarrow &= A(E_1\downarrow,E_2,E_3)\downarrow \\ \neg(E_2)[E_1,E_3]\downarrow &= \neg(E_2)[E_1\downarrow,\uparrow E_3]\downarrow\end{aligned}$$

Crucially, it is E_2, and not $E_2\downarrow$, which is required on the right-hand side in each case.[3]

7 Conclusions and Further Work

We have drawn a contrast between two styles of event definition: the active database approach in which events are regarded as instantaneous and defined in terms of the conditions for their detection at an instant, and the knowledge representation approach in which events are for the most part regarded as durative and are defined in terms of the conditions for their occurrence over an interval. The contrast is starkly drawn in order to make a point; it is not suggested that there has been *no* dialogue and no commonality between the two approaches. We have explored this contrast in some detail by examining SNOOP, a system proposed within the active database community, displaying some of its shortcomings, and showing that these can be rectified by remodelling

[3] This gives the finishing time for $\neg(E_2)[E_1, E_3]$; but this event cannot be *detected* until the time of $E_3\downarrow$—the only mismatch between time of detection and time of finishing, a subtlety that doesn't arise for instantaneous events.

the system in the knowledge representation style. We further showed that SNOOP in its original form can be partially, but not totally, recreated within the remodelled system by mapping each event onto its associated detection event.

We regard this result as establishing a preliminary bridge between the approaches to events espoused by the two communities. To make further progress it will be necessary, from the active database side, to investigate the effect on their applicability of the remodelling of the event-forming operators, and, on the knowledge representation side, to investigate the relation of the SNOOP-style operators to the event-constructors already in common use in knowledge representation contexts. We believe that there is scope for fruitful dialogue between the two communities in these endeavours.

Acknowledgments

We should like to thank Brian Lings and Jonas Mellin for useful feedback on this paper, and Pernilla Rönn for pointing out an error in our original definition of $\triangle$.

References

1. N. Gehani, H. Jagadish, and O. Shmueli. Event specification in an active object-oriented database. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 81–90, San Diego, Calif., 1992.
2. Stella Gatziu and Klaus R. Dittrich. Events in an active object-oriented database system. In *Proc. 1st Int. Conf. on Rules in Database Systems*, Edinburgh, 1993.
3. Stella Gatziu and Klaus R. Dittrich. Detecting composite events in active database sytems using Petri nets. In *Proc. 4th Int. Workshop on Research Issues in Data Engineering: Active Database Systems*, pages 2–9, Edinburgh, 1994.
4. S. Chakravarty, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active database: Semantics, contexts, and detection. In *20th International Conference on Very Large Databases*, pages 606–617, Santiago, Chile, September 1994.
5. Elisa Bertino, Elena Ferrari, and Giovanna Guerrini. An approach to model and query event-based temporal data. In *Proceeedings of the Fifth International Workshop on Temporal Representation and Reasoning (TIME'98)*, pages 122–131. IEEE Computer Science Press, 1998.
6. Claudia L. Roncancio. Towards duration-based, constrained and dynamic event types. In Sten F. Andler and Jörgen Hansson, editors, *Active, Real-Time, and Temporal Database Systems (Proc.2nd Int. Workshop ARTDB-97, Como, Italy, September 1997)*, pages 176–193. Springer-Verlag, 1997.
7. James Allen. Towards a general theory of action and time. *Artif. Intell.*, 23:123–54, 1984.
8. James Allen and George Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4:531–79, 1994.
9. R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.
10. Antony P. Galton. Space, time and movement. In Oliviero Stock, editor, *Spatial and Temporal Reasoning*, pages 321–352. Kluwer Academic Publishers, Dordrecht, 1997.
11. Malik Ghallab. On chronicles: Representation, on-line recognition, and learning. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *Principles of Knowledge Representation and Reasoning (Proceedings of KR'96)*, pages 597–606, San Francisco, CA, 1996. Morgan Kaufmann.