Durative Events in Active Databases

Rodolfo Gómez[*]

Computing Laboratory
University of Kent at Canterbury
CT2 7NF, Canterbury, Kent, United Kingdom
rsg2@kent.ac.uk

Juan Carlos Augusto

School of Computing and Mathematics
University of Ulster at Jordanstown
BT37 0QB Newtownabbey, United Kingdom
jc.augusto@ulster.ac.uk

Key words: Active databases, event specification language, durative events, composition operators, composition semantics.

Abstract: Active databases are DBMS which are able to detect certain events in the environment and trigger actions in consequence. Event detection has been subject of much research, and a number of different event specification languages is extant. However, this is far from being a trivial or accomplished task. Most of these languages handle just instantaneous events, but it has been noticed that a number of situations arise where it would be interesting or even necessary to handle durative events. We elaborate on a given specification language which combines instantaneous and durative events, revealing some issues which must be taken into account when the semantics of event composition is defined.

1   INTRODUCTION

Active Databases (Paton and Diaz, 1999) are database management systems (DBMS) which are able to perform actions in response to the detection of particular events. The active behavior of these systems is usually defined through *E-C-A* rules (*Event - Condition - Action*) (Berndtsson and Lings, 1995). Research has been devoted to both models and languages for events (see e.g., (Gehani et al., 1992b; Gehani et al., 1992a; Chakravarthy and Mishra, 1993; Gatziu and Dittrich, 1994; Galton, 1995; Roncancio, 1999; Galton, 2000; Gómez et al., 2000; Gómez et al., 2001; Galton and Augusto, 2002)). Generally, events are classified as *primitive* if they can be detected directly, e.g., deleting a tuple; or as *composite* if they are higher level constructs expressing some relationship between more primitive events, e.g., deleting a sequence of tuples. Events defined in database contexts are useful for analyzing the history of a given operation or querying database states. From (Galton and Augusto, 2002) we learnt that research considering events in the context of active databases is far from being either a trivial or an accomplished task. Most of the models and prototypes proposed so far have considered instantaneous events only, and little attention has been given to the consideration of dura-

tive events. Nevertheless, it has been recognized that duration provides a better semantics for event composition (Galton, 2000). Also, certain database operations are more naturally modelled by durative events (Roncancio, 1999). The work in (Roncancio, 1999) is one of the few attempts made so far to combine both instantaneous and durative events in the same language. As it is usual in event languages, a set of primitive events can be composed into more complex structures by using a number of composition operators. We have found, however, that some issues concerning the definition and detection of composite events have not been satisfactorily identified, leading to a number of problems which may arise in the use of some of the composition operators. We show how these operators can be modified in such a way that consistency is preserved through the operator set. We also discuss the effect of different composition semantics for durative events. We believe this discussion is another necessary step in the direction suggested in (Galton and Augusto, 2002) to clarify some fundamental notions which underly the use of events in active databases. It is important to stress that the language proposed in (Roncancio, 1999) is considered as the reference language for durative events in the technical literature of active databases.

Section 2 will provide the main concepts behind the event language proposed in (Roncancio, 1999). In section 3 problems arising in the use of some compo-

---

sition operators, their causes and undesired effects are identified. Section 3.1 shows how these operators can be redefined to avoid the problems previously mentioned. In section 3.2 we discuss some different semantics which can be assigned to composition operators. Conclusions are given in section 4. More comprehensive discussions and other problems are given in the full version of this article (Gómez and Augusto, 2003).

2 A DURATIVE EVENT SPECIFICATION LANGUAGE

Roncancio (Roncancio, 1999) proposes to extend the instantaneous event model of NAOS (Collet and Coupaye, 1996), an event detector module which is part of the object-oriented DBMS O2 (Bancilhon et al., 1992). We will refer to the proposal made in (Roncancio, 1999) as E-NAOS (for Extended NAOS) from now on. ECA rules take the general form:

```
on <event_expression>
if <condition>
do <action>
```

A durative event identifies a "happening of interest" which occurs over an interval of time. A durative event can be seen as an abstraction constructed over two instantaneous events which bound its occurrence period. Durative events can, for example, be related to database operations like updating a tuple stored in the database, which can be more naturally modelled as having a related duration (instead of handling such start-end instantaneous events (Roncancio, 1999)). Also, durative events provide convenient semantics to handle composite events (Galton, 2000).

An event type describes a set of instances with the same behavior, e.g., tuple insertion. An event instance carries some information related to its occurrence. Some information depends on the event type, e.g., the values of every field in a tuple insertion, but other information is common to all types. Our main concern will be the time of occurrence and the time of detection. For instantaneous events these two times usually meet. For durative events, however, the time of occurrence (called *occurrence period*) denotes the span which bounds the event instance while the time of detection (called *notification time*) is an instant equal or greater than the last instant of the occurrence period. This model consider that instantaneous events are durative events with a minimum duration (a *chronon*, see (Jensen et al., 1992)), so they also have an occurrence period (although minimal).

Events are classified as *primitive* or *composite*. Primitive events are related, amongst other things, to read-write operations on objects, method calls and transaction executions.

Composite events are defined by composing primitive events or by composing other composite events, using a set of operators. Operators are based on the classic Allen's interval relations (Allen, 1983), and on some instantaneous operators defined in (Chakravarthy et al., 1994). Composition results in a new durative event with its own occurrence period. Operators are described below (Table 1 and Fig. 1), but first, a description of the notation is in order. A, B, etc. denote events. OP and NT stand for *occurrence period* and *notification time*, respectively. Functions *max* and *min* return the greatest and least instant of a pair, respectively. E^-, E^+ stand for the bounds of the occurrence period of E (i.e. $OP=[E^-,E^+]$). Finally, note that we have assumed that the disjunction operator (or) is exclusive. This is not clear from (Roncancio, 1999), but it seems to be the proper semantics given the context. Also, the NT given for overlaps corresponds to the first occurrence condition (the NT is symmetric if the other condition occurs). In any case, these do not affect the results shown in this paper.

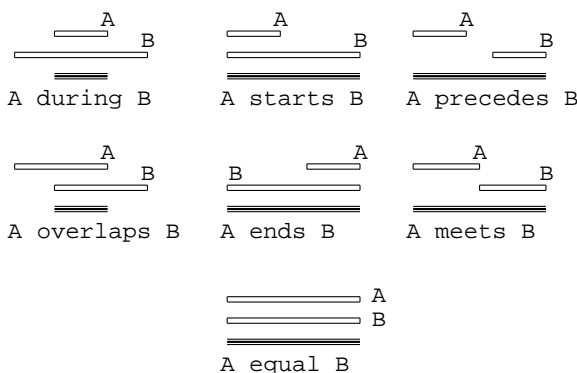Fig. 1 shows (in boldface) the occurrence periods for the relational operators.

Figure 1: Occurrence periods

Notice that the occurrence periods are not consistently defined for all operators. In some of them (during and overlaps) the occurrence period only comprises the interval of time when the two components are simultaneously occurring, whereas in the other operators the period covers both components. Also, notification times are sometimes defined as the last instant of the occurrence period (precedes, starts, ends) while for others (overlaps and during) it depends on whether the component events are primitive or composite.

Operator	Definition
A precedes B:	$\text{A}^+ < \text{B}^-$
OP:	$[\text{A}^-,\text{B}^+]$
NT:	B^+
A during B:	$\text{B}^- < \text{A}^- < \text{A}^+ < \text{B}^+$
OP:	$[\text{A}^-,\text{A}^+]$
NT:	A^+ if B is primitive
	B^+ if B is composite
A overlaps B:	$\text{A}^- < \text{B}^- < \text{A}^+ < \text{B}^+$ or
	$\text{B}^- < \text{A}^- < \text{B}^+ < \text{A}^+$
OP:	$[max(\text{A}^-,\text{B}^-),min(\text{A}^+,\text{B}^+)]$
NT:	A^+ if B primitive
	B^+ if B is composite
A starts B:	$\text{A}^- = \text{B}^-$
OP:	$[\text{A}^-,max(\text{A}^+,\text{B}^+)]$
NT:	$max(\text{A}^+,\text{B}^+)$
A equal B:	$\text{A}^- = \text{B}^- \wedge \text{A}^+ = \text{B}^+$
OP:	$[\text{A}^-,\text{A}^+]$
NT:	A^+
A ends B:	$\text{A}^+ = \text{B}^+$
OP:	$[min(\text{A}^-,\text{B}^-),\text{A}^+]$
NT:	A^+
A meets B:	$\text{A}^+ + 1\ chronon = \text{B}^-$
OP:	$[\text{A}^-,\text{B}^+]$
NT:	B^+
A or B:	either A or B occurs (but not both)
OP:	$OP(\text{A})$ if A occurs
	$OP(\text{B})$ if B occurs
NT:	$NT(\text{A})$ if A occurs
	$NT(\text{B})$ if B occurs
A and B:	both A and B occur
OP:	$[min(\text{A}^-,\text{B}^-),max(\text{A}^+,\text{B}^+)]$
NT:	$max(\text{A}^+,\text{B}^+)$

Table 1: Composition operators in E-NAOS

3 SOME UNDESIRED EFFECTS OF A COMPOSITION SEMANTICS

This section presents an example which reveals some problems with the composition operators proposed for E-NAOS (others can be found in (Gómez and Augusto, 2003)). We will see a number of situations where the detection of composite events may have an unexpected (and possibly incorrect) outcome. These problems may not arise in the database context the language is currently used (i.e. NAOS-O2). Nevertheless, our intention is to show that some unsafe expressions may result when composition of durative events is not carefully defined.

Consider a laboratory of a given pharmaceutical company, testing the effects of new drugs on cells infected with some virus. Incompatibilities be-

tween different drugs, unwanted side effects, amongst other hazardous situations, are part of the information which is expected from the tests. We can imagine the use of an active DBMS in such a context. For example, the current level of certain substances in the cells can be detected as an indication of some situation of interest, which in turn can be thought of as primitive, detectable events. Different actions can be triggered, e.g. an alarm signal or the modification of the cell's environment, such as temperature or moisture conditions. Therefore, we can assume a number of ECA rules in place, and importantly, that durative events must be handled (reactions in the cell may have a related duration).

In (Roncancio, 1999), the occurrence period for operators `overlaps` and `during` only comprises the period when the two component events are simultaneously occurring, whereas for every other operator the period covers both component occurrences (see Fig. 1). This inconsistency may cause some problems, as shown in Example 1.

EXAMPLE **1** *The following active rule detects a given reaction in the cell (event* R*) under the effect of two drugs (events* A *and* B*). Suppose that this reaction is meaningless unless a) it is detected after the first drug and before the second drug have made effect on the cell and b) the effect of the first drug overlaps or precedes the effect of the second drug. The rule below appears, then, as a natural solution:*

```
on R during
        (   (A overlaps B)
        or (A precedes B))
do (...)
```

Figure 2 shows two situations in which the rule should have been triggered. However, and because occurrence periods are assigned differently to overlaps *and* precedes*, the reaction is not detected when the effects of both drugs overlap, even when it happens before the effect of the second drug (*B*) is gone (case 1).*
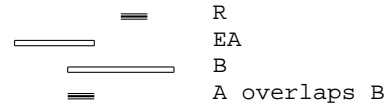
case 1: rule not triggered!
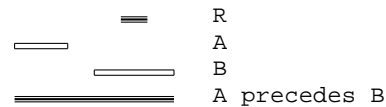
case 2: rule triggered

Figure 2: Missing detections (E-NAOS ops.)

Other kind of problem arises because notification times are not uniformly defined. In (Roncancio, 1999), composite events `during` and `overlaps` are notified to the system before all component occurrences have been detected. For example, `A during B` is notified when `A` finishes if `B` is primitive, but instead it is notified when `B` finishes if `B` is composite. Example 2 shows a possible consequence of this definition.

EXAMPLE **2** *The following rule detects a reaction (*R*) in a cell under the effects of two drugs (*A *and* B*). Different tests include the administration of a single drug or both drugs together, and the reaction is considered meaningless unless the effects of drugs are completely detected (even though they can be considered as primitive events, complete detection cannot always be ensured). Such a rule, then, could be written as follows:*

```
on R during
     (A or B or (A and B))
do (...)
```

Fig. 3 shows three different situations; when the reaction is detected with respect to either A *or* B*, the rule is triggered immediately afterwards. This is not safe as* A *or* B *have not been completely detected at that moment, and so it can be the case that the reaction is meaningless if such occurrences fail to happen. On the other hand, if both drugs are administered the rule will not be triggered unless both* A *and* B *have been completely detected (case 3).*

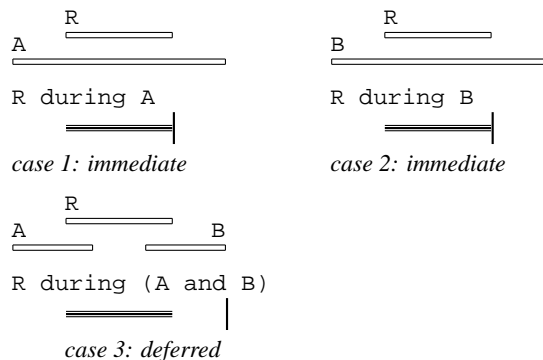case 1: immediate   case 2: immediate

case 3: deferred

Figure 3: Deferred triggering using operators proposed for E-NAOS

3.1 Composition Operators With A Consistent Semantics

Definitions can be modified to obtain a new set of composition operators where a) the occurrence period of the resulting composite event includes the complete occurrences of the components, and b) no com-

posite event is notified until all components have been notified. Moreover, the notification time of a composite event is the same whether components are primitive or composite. Therefore, problems shown in examples 1 and 2 no longer arise. The modified operators are shown in Table 2 and Fig. 4. Operators have been also modified to follow the original definitions of Allen's interval relations. Pragmatically, this is desirable since Allen's relations have a well-known semantics that users may naturally expect when using operators with similar names. The full paper (Gómez and Augusto, 2003) shows an example of how the semantics of E-NAOS operators could be misinterpreted and lead to unexpected behaviour.

Operator	Definition
A precedes' B:	$A^+ < B^-$
OP:	$[A^-, B^+]$
NT:	B^+
A during' B:	$B^- < A^- < A^+ < B^+$
OP:	$[B^-, B^+]$
NT:	B^+
A overlaps' B:	$A^- < B^- < A^+ < B^+$
OP:	$[A^-, B^+]$
NT:	B^+
A starts' B:	$A^- = B^-$
OP:	$[A^-, B^+]$
NT:	B^+
A equal' B:	$A^- = B^- \wedge A^+ = B^+$
OP:	$[B^-, B^+]$
NT:	B^+
A ends' B:	$A^+ = B^+$
OP:	$[B^-, B^+]$
NT:	B^+
A meets' B:	$A^+ + 1$ *chronon* $= B^-$
OP:	$[A^-, B^+]$
NT:	B^+

Table 2: Modified operators

A during' B   A starts' B   A precedes' B

A overlaps' B   A ends' B   A meets' B
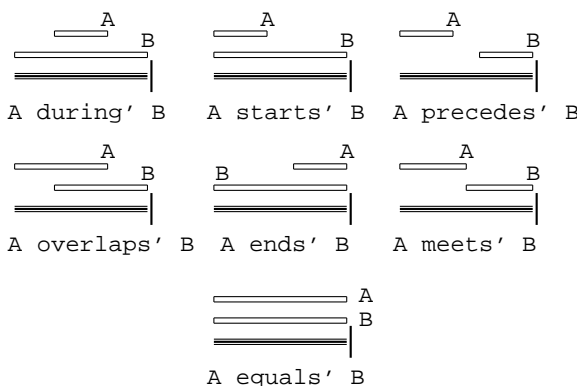
A equals' B

Figure 4: Occurrence periods (modified operators)

4

3.2 A Discussion On Composition Semantics

We would like to emphasize that the potential problems found in E-NAOS do not arise because of a particular set of operators was chosen (this depends on the context) but because the set is not consistent. Operators in E-NAOS have been overloaded with different semantics, where each one of these is only adequate when the operator is used in a specific context. For example, all Allen-like operators can be used to constrain the conjunction of two events, i.e. the occurrences of both components have been detected, and these satisfy certain temporal placement. For all operators but `during` and `overlaps`, though, the resulting event has an occurrence period which covers both components. On the other hand, `overlaps` and `during` results in an event whose occurrence period only covers the shared interval between components. Sometimes, this may be convenient. The following rule triggers only if event A occurs while B is occurring:

on A during B **do** (...)

If event B was used just to constrain A, then in the action part of rule we will probably be concerned only with the occurrence period of A. Hence, there is no reason for operator `during` to be assigned an occurrence period that includes both component events. However, we have shown in Example 1 that sometimes conjunction semantics is more convenient. But there is, a priori, no reason why the occurrence period for `during` and `overlaps` should not cover both components (as is the case for all other operators in the same class).

Some composite events in E-NAOS can be detected before all component events have been fully detected, e.g., `during` and `overlaps`. Sometimes this *eager detection* may be convenient, but in other contexts a more safe approach could be required (Example 2). Again, consistency is more important than particular contexts, i.e., for example, the notification time for `starts` should not be different than the one assigned to `during`.

4 CONCLUSIONS

We analyze in this work a well-known proposal for representing and reasoning with durative events in active databases. We identify a number of problems which may arise in event specification languages when the semantics assigned to the composition of durative events is not consistent. We illustrate such

problems using the event language presented in (Roncancio, 1999) as a case study. We show how apparently innocuous definitions results in operators with undesired side effects and behaviours which are difficult to predict. This, in consequence, results in a language where "unsafe" expressions are not easy to discover. We also show how a simple modification of these operators achieves consistency through the set, thus ruling out the possibly unsafe situations. The full paper (Gómez and Augusto, 2003) shows other contexts where composition operators in E-NAOS present an undesired behaviour; also, a more comprehensive discussion about possible semantics for that kind of operators is offered.

As (Galton and Augusto, 2002) and this work have shown, neither the proposals given in (Chakravarthy et al., 1994) nor in (Roncancio, 1999) are free from problems in their attempts to accommodate durative events in different ways. In this article we gave another step on raising awareness of the important remaining problems. Despite the importance of the topic, there is not satisfactory proposal in the area and much more work is still needed to clarify fundamental notions underlying the use of events in active database systems. We expect the problems discussed in this paper will inspire new proposals towards a next generation of more reliable systems.

REFERENCES

Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832 – 843.

Bancilhon, F., Delobel, C., and P. Kanellakis, editors (1992). *Building an Object-Oriented Database System - The Story of O2*. Morgan Kaufmann.

Berndtsson, M. and Lings, B. (1995). Logical Events and ECA Rules. Technical Report HS-IDA-TR-95-004, Department of Computer Cience, University of Skövde.

Chakravarthy, S., Krishnaprasad, V., Anwar, E., and Kim, S. (1994). Composite Events for Active Databases: Semantics, Contexts and Detection. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'94)*, pages 606–617, Santiago de Chile, Chile.

Chakravarthy, S. and Mishra, D. (1993). Snoop: An expressive Event Specification Language for Active Databases. Technical Report UF-CIS-TR-93-007, University of Florida, USA.

Collet, C. and Coupaye, T. (1996). Composite Events in NAOS. In *7th International Conference and Workshop on Database and Expert Systems Applications (DEXA'96). LNCS 1134*, pages 244–253, Zurich, Switzerland.

Galton, A. (1995). Time and Change. In *Hanbook of Logic in Artificial Intelligence and Logic Programming*, volume 4 (Epistemic and Temporal Reasoning), pages 175–240. D. Gabbay and C. Hogger and J. Robinson (eds.), Clarendon Press.

Galton, A. (2000). Eventualities. In Vila, van Beek, Boddy, Fisher, Gabbay, Galton, and Morris, editors, *The Handbook of Time and Temporal Reasoning in Artificial Intelligence*. MIT Press. (to be published).

Galton, A. and Augusto, J. C. (2002). Two approaches to event definition. In A. Hameurlain, R. C. and Traunmüller, R., editors, *Proceedings of 13th International Conference on Database and Expert Systems Applications (DEXA 2002), Aix-en-Provence, France, Springer-Verlag*, pages 547–556.

Gatziu, S. and Dittrich, K. (1994). Detecting Composite Events in Active Databases Systems using Petri Nets. In *Proceedings of the 4th International Workshop on Research Issues in Data Engineering: Active Database Systems*, pages 2–9, Houston, USA. IEEE.

Gehani, N., Jagadish, H. V., and Shmueli, O. (1992a). Composite Event Specification in Active Databases: Model & Implementation. In *Proceedings of the 18th International Conference on Very Large Data Bases (VLDB'92)*, pages 327–338.

Gehani, N., Jagadish, H. V., and Shmueli, O. (1992b). Event Specification in an Active Object-Oriented Database. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, pages 81–99, San Diego, California, USA.

Gómez, R. and Augusto, J. (2003). Durative Event Composition in Active Databases. Technical report, University of Ulster at Jordanstown. (http://www.infj.ulst.ac.uk/~jcaug/gomezaug.pdf).

Gómez, R., Augusto, J. C., and Galton, A. (2000). Implementation and Testing for a Set of Event Detection Operators. Technical Report 398, School of Engineering and Computer Science, University of Exeter, United Kingdom. (http://www.infj.ulst.ac.uk/~jcaug/rr398.pdf).

Gómez, R., Augusto, J. C., and Galton, A. (2001). Testing an Event Specification Language. In *Proceedings of the 13th. International Conference of Software Engineering and Knowledge Engineering (SEKE 2001)*, pages 341–346, Buenos Aires, Argentina.

Jensen, C., Clifford, J., Gadia, S., Segev, A., and Snodgrass, R. (1992). A Consensus Glossary of Temporal Database Concepts. *SIGMOD Record*, 21, num. 3.

Paton, N. and Diaz, O. (1999). Active Database Systems. *ACM Computing Surveys*, 31(1):63–103.

Roncancio, C. L. (1999). Toward Duration-Based, Constrained and Dynamic Event Types. In *Proceedings of the Second International Workshop on Active, Real-Time and Temporal Database Systems (ARTDB'97), Como, Italy, September 8-9, 1997*, number 1553 in LNCS, pages 176–194. Sten F. Andler and Jörgen Hansson (eds.), Springer.