

# Composite Events for Active Databases

Semantics, Contexts and Detection

*Chakravarthy, Krishnaprasad, Anwar, Kim*

Presentation: Fabio Zünd 2007

# Content

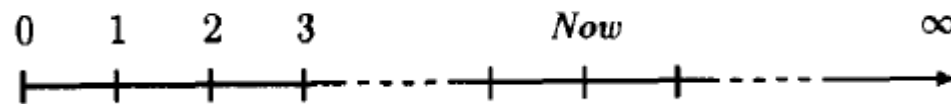
- Introduction
- Snoop: An event specification language
- Sentinel: Active OODBMS
- Conclusion

# Introduction

- ECA Rule = (Event, Condition, Action)
- Need expressive event specification language

# Semantics of Snoop

- Timeline

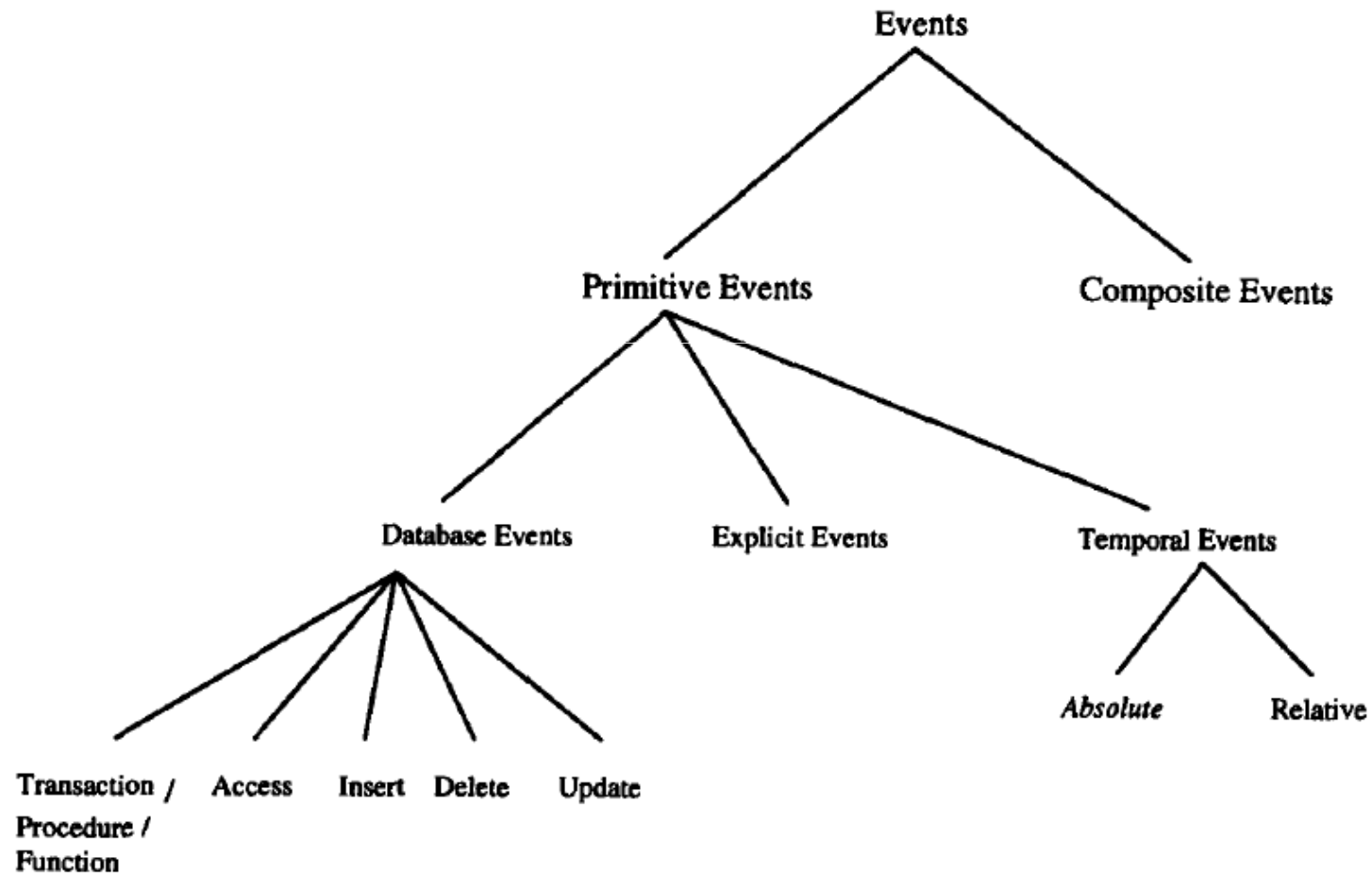


- Event
- Event Expression
- Event Modifiers

# Snoop: Event

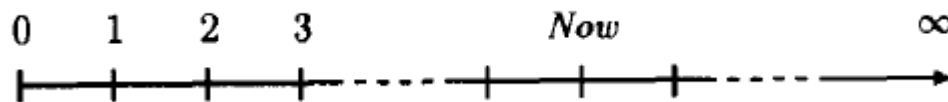
- Instantaneous, atomic occurrence of interest at a point in time
- Types of events

# Snoop: Events



# Snoop: Event Expression

- Interval on timeline



# Snoop: Event Modifiers

- Transform event expressions into events
- *begin\_of, end\_of*

# ECA Rule Example

- „A withdrawal following a deposit is permitted for a fixed amount irrespective of the account balanced“

R1 :	ON	deposit
	CONDITION	.....
	ACTION	set deposit:=true; deactivate R1; activate R2;
R2 :	ON	withdraw
	CONDITION	deposit=true
	ACTION	set deposit:=false; allow_withdraw

# Snoop: Event Operators

- An event is a function

$$E : T \rightarrow \{\text{True}, \text{False}\}$$

$$E(t) = \begin{cases} \text{T(rue)} & \text{if an event of type } E \text{ occurs} \\ & \text{at time point } t \\ \text{F(alse)} & \text{otherwise.} \end{cases}$$

- Negation  $\sim E$

# Snoop: Event Operators 2

- **OR:**  $E_1 \vee E_2$
- **AND:**  $E_1 \wedge E_2$
- **ANY:**  $ANY(m, E_1, E_2, \dots, E_n)$ 
  - Occurs when m events out of n distinct events occur

# Snoop: Event Operators 3

- **SEQ:**  $E_1; E_2$
- **A:**  $A(E_1, E_2, E_3)$ 
  - Aperiodic Operator (non-cumulative)
  - Occurs **each time** E2 occurs within the time interval {E1,E3}
- **A\*:**  $A^*(E_1, E_2, E_3)$ 
  - Aperiodic Operator (cumulative)
  - Occurs **only once** when E2 occurs within the time interval {E1,E3}

# Snoop: Event Operators 4

- **P**:  $P(E_1, TI[ :parameters ], E_3)$ 
  - Periodic Event Operator
  - Occurs **for every** TI interval, starting after E1 and ceasing after E3
- **P\***:  $P^*(E_1, TI :parameters, E_3)$ 
  - Aperiodic Operator (cumulative)
  - Occurs **only once** when E3 occurs
- **NOT**:  $\neg (E_2)[E_1, E_3]$ 
  - detects non-occurrence of E2 in closed interval

# Snoop: Rule Examples

- When 4 withdrawals are made on an account in a day, do not allow further withdrawals after that day.

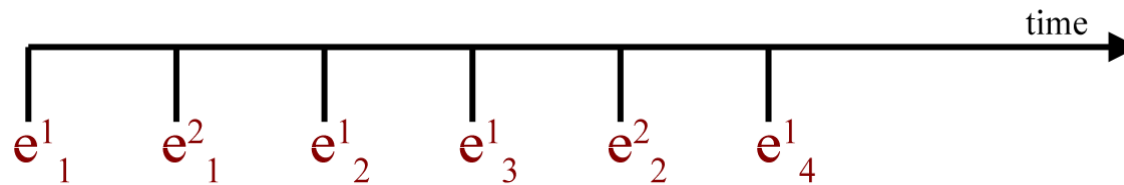
```
On           A(8 a.m, ANY(4,withdraw-on-account*), 5 p.m.)
Condition    true
Action       block further withdrawals
```

- If after opening of stock market ( $E_1$ ) and any change in Dow Jones average ( $E_2$ ) a change in the price of IBM stock ( $E_3$ ) follows and after that again  $E_2$  and a change in a commodity witch depends on IBM stock ( $E_4$ ) happens, apply some action.

```
On           (E1 $\Delta$ E2) ; E3 ; (E2 $\Delta$ E4)
Condition    true
Action       some_action
```

# Histories and Event Logs

- Occurrence of event type  $E_j$  denoted by  $e_j^i$

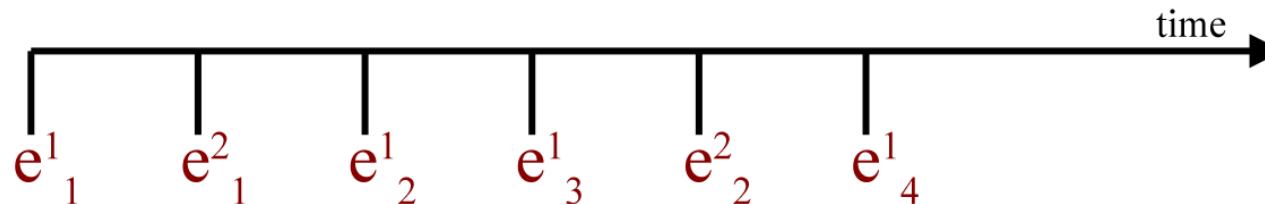


- **Global Event History H**
  - set of all primitive event occurrences

# Histories and Event Logs 2

- **Primitive Event History**  $E[H]$  of a primitive event type  $E$
- **Composite Event History**  $E[H]$  of a composite event  $E$
- **Event Collection**  $p(E, start\_time, end\_time)$

# Unrestricted Context



- $(E_1 \Delta E_2)[H] = \{ \{e^i, e^j\} \mid \{e^i, e^j\} \in ((E_1[H] \uplus E_2[H]) \cup (E_2[H] \uplus E_1[H])) \text{ and } t_{\text{occ}}(e^i) \leq t_{\text{occ}}(e^j) \}$ .
- $(E_1; E_2)[H] = \{ \{e^i, e^j\} \mid t_{\text{occ}}(e^i) < t_{\text{occ}}(e^j) \text{ and } \{e^i, e^j\} \in E_1[H] \uplus E_2[H] \}$ .

# Unrestricted Context

$$X = ((E_1 \Delta E_2); E_3; (E_2 \Delta E_4))$$

$$H = \{\{e_1^1\}, \{e_1^2\}, \{e_2^1\}, \{e_3^1\}, \{e_2^2\}, \{e_4^1\}, \{e_3^2\}, \{e_4^2\}\}$$

$$E_1[H] = \{\{e_1^1\}, \{e_1^2\}\}$$

$$E_2[H] = \{\{e_2^1\}, \{e_2^2\}\}$$

$$E_3[H] = \{\{e_3^1\}, \{e_3^2\}\}$$

$$E_4[H] = \{\{e_4^1\}, \{e_4^2\}\}$$

$$(E_1 \Delta E_2)[H] = \{\{e_1^1, e_2^1\}, \{e_1^1, e_2^2\}, \{e_1^2, e_2^1\}, \{e_1^2, e_2^2\}\}$$

$$(E_2 \Delta E_4)[H] = \{\{e_2^1, e_4^1\}, \{e_2^1, e_4^2\}, \{e_2^2, e_4^1\}, \{e_2^2, e_4^2\}\}$$

$$X[H] = \{\{e_1^1, e_2^1, e_3^1, e_2^1, e_4^1\}, \{e_1^1, e_2^1, e_3^1, e_2^1, e_4^2\}, \\ \{e_1^1, e_2^1, e_3^1, e_2^2, e_4^1\}, \{e_1^1, e_2^1, e_3^1, e_2^2, e_4^2\}, \\ \{e_1^1, e_2^1, e_3^2, e_2^1, e_4^1\}, \{e_1^1, e_2^1, e_3^2, e_2^1, e_4^2\}, \\ \{e_1^1, e_2^1, e_3^2, e_2^2, e_4^1\}, \{e_1^1, e_2^1, e_3^2, e_2^2, e_4^2\}, \\ \{e_1^2, e_2^1, e_3^1, e_2^1, e_4^1\}, \{e_1^2, e_2^1, e_3^1, e_2^1, e_4^2\}, \\ \{e_1^2, e_2^1, e_3^1, e_2^2, e_4^1\}, \{e_1^2, e_2^1, e_3^1, e_2^2, e_4^2\}, \\ \{e_1^2, e_2^1, e_3^2, e_2^1, e_4^1\}, \{e_1^2, e_2^1, e_3^2, e_2^1, e_4^2\}, \\ \{e_1^2, e_2^1, e_3^2, e_2^2, e_4^1\}, \{e_1^2, e_2^1, e_3^2, e_2^2, e_4^2\}, \\ \{e_1^2, e_2^2, e_3^2, e_2^1, e_4^1\}, \{e_1^2, e_2^2, e_3^2, e_2^1, e_4^2\}, \\ \{e_1^2, e_2^2, e_3^2, e_2^2, e_4^1\}, \{e_1^2, e_2^2, e_3^2, e_2^2, e_4^2\}\}$$

# Composite Event Detection

- Parameter contexts to detect and compute parameters of composite events in different ways

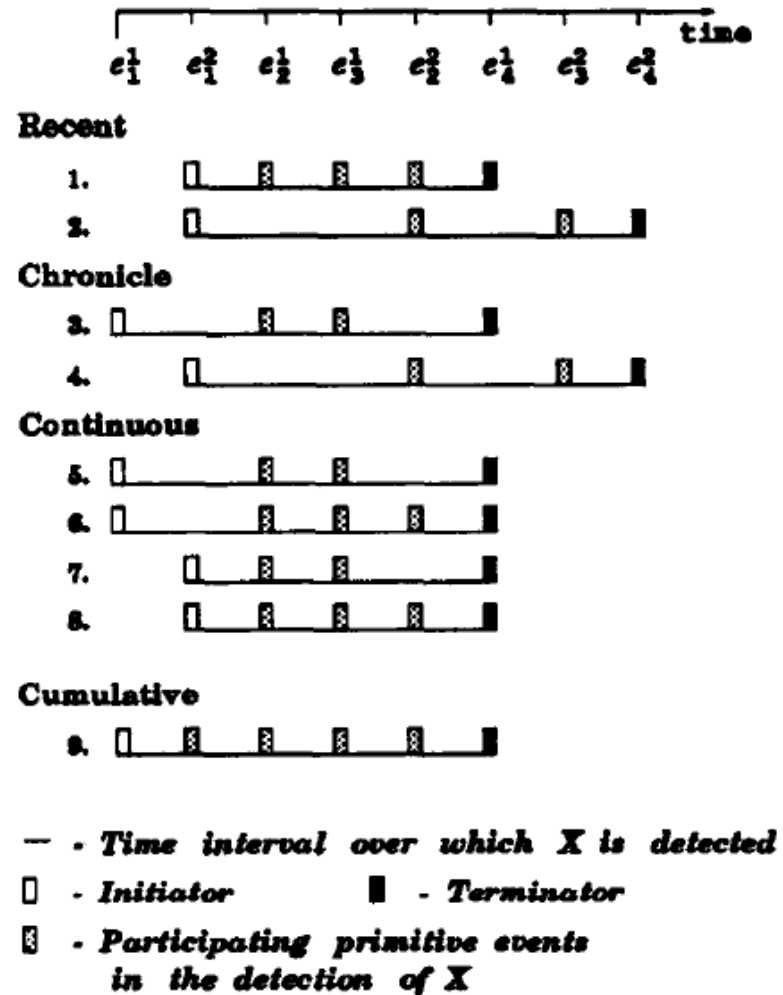
# Parameter Contexts 1

- **Recent**

- Only the **most recent** occurrence
- Global tracking system

- **Chronicle**

- Initiator, terminator pair s unique
- Transactions



$$X = (E_1 \Delta E_2; E_3; E_2 \Delta E_4)$$

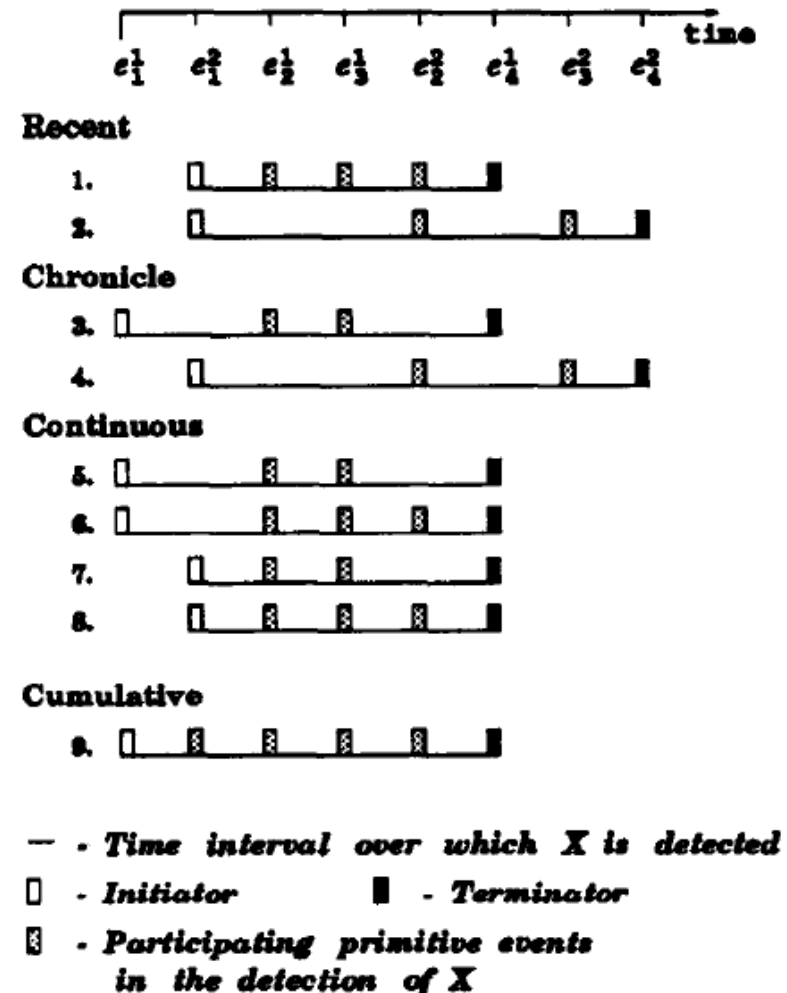
# Parameter Contexts 2

- **Continuous**

- Each initiator starts the detection
- Forecasting

- **Cumulative**

- For each constituent event, all occurrences of the event are accumulated until detection
- Deadline event

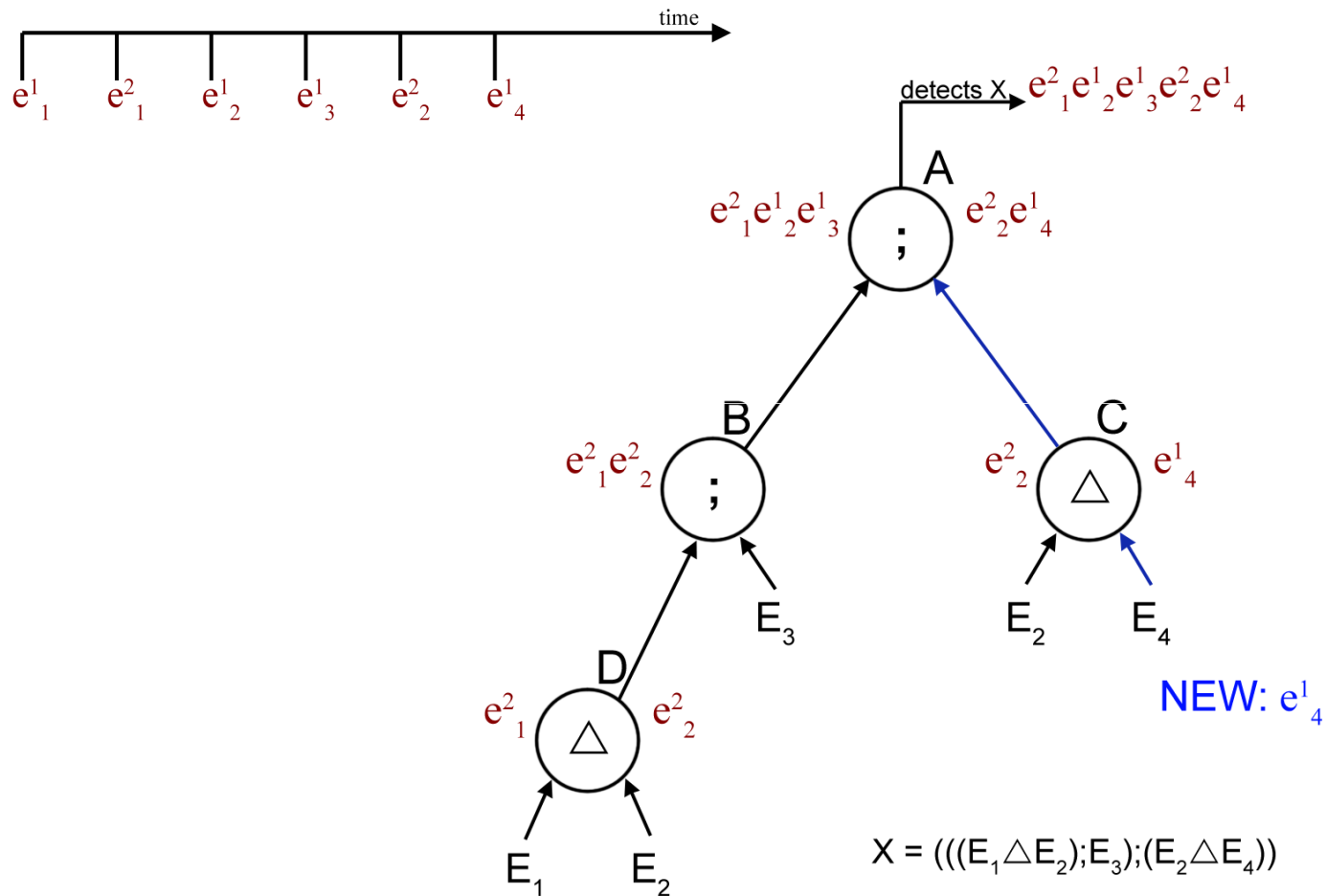


$$X = (E_1 \Delta E_2; E_3; E_2 \Delta E_4)$$

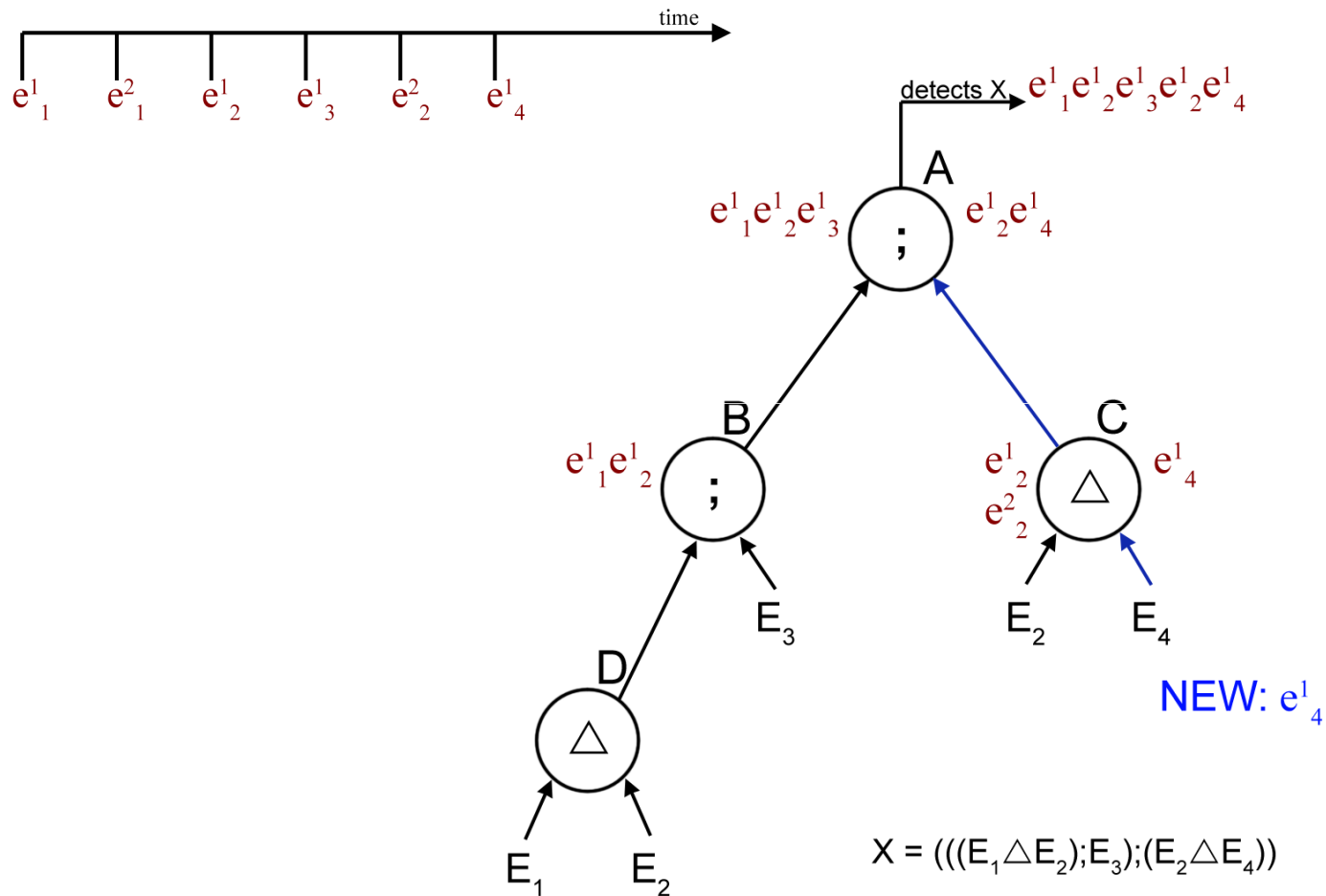
# Composite Event Detection

- Event tree

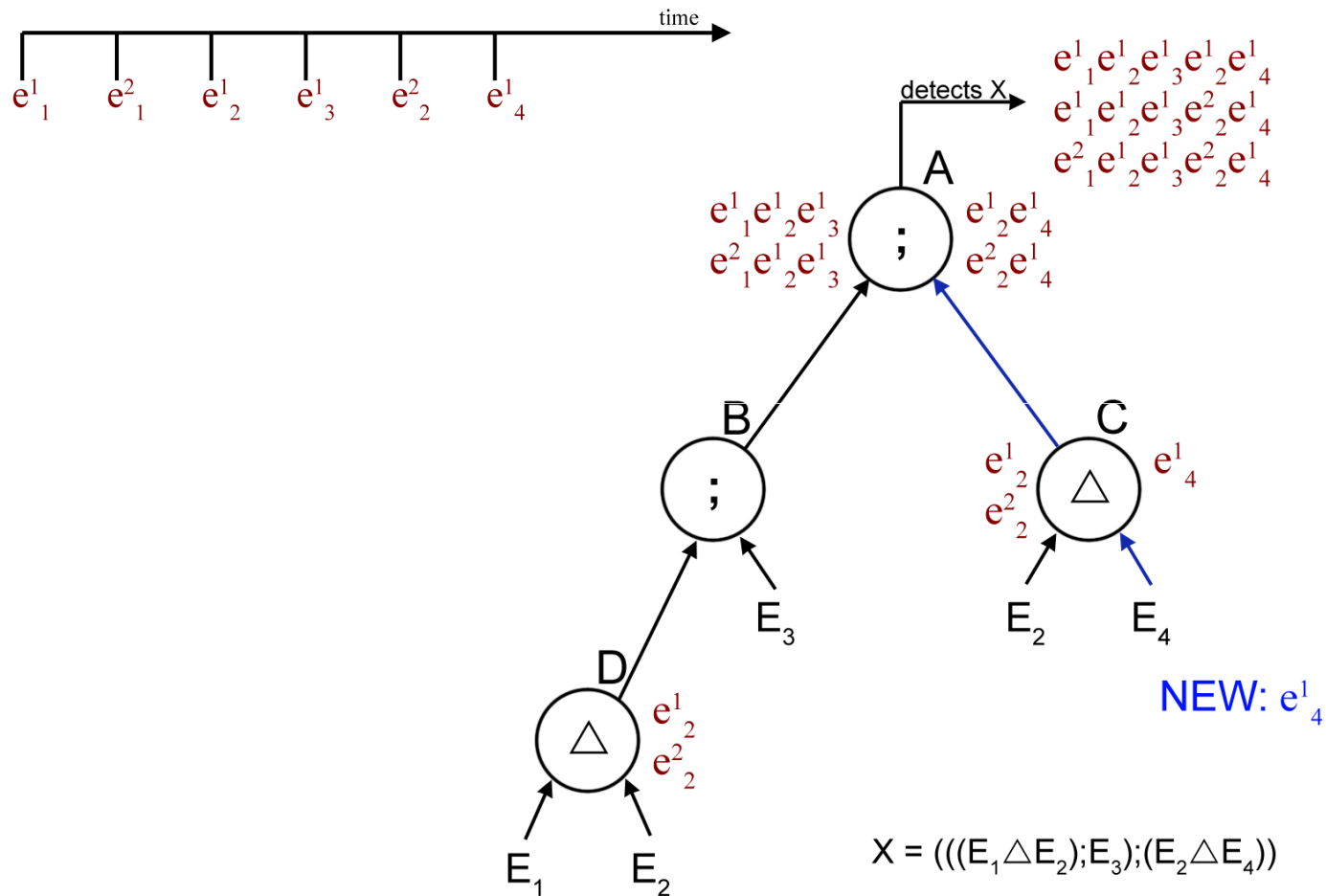
# CED: Recent Context



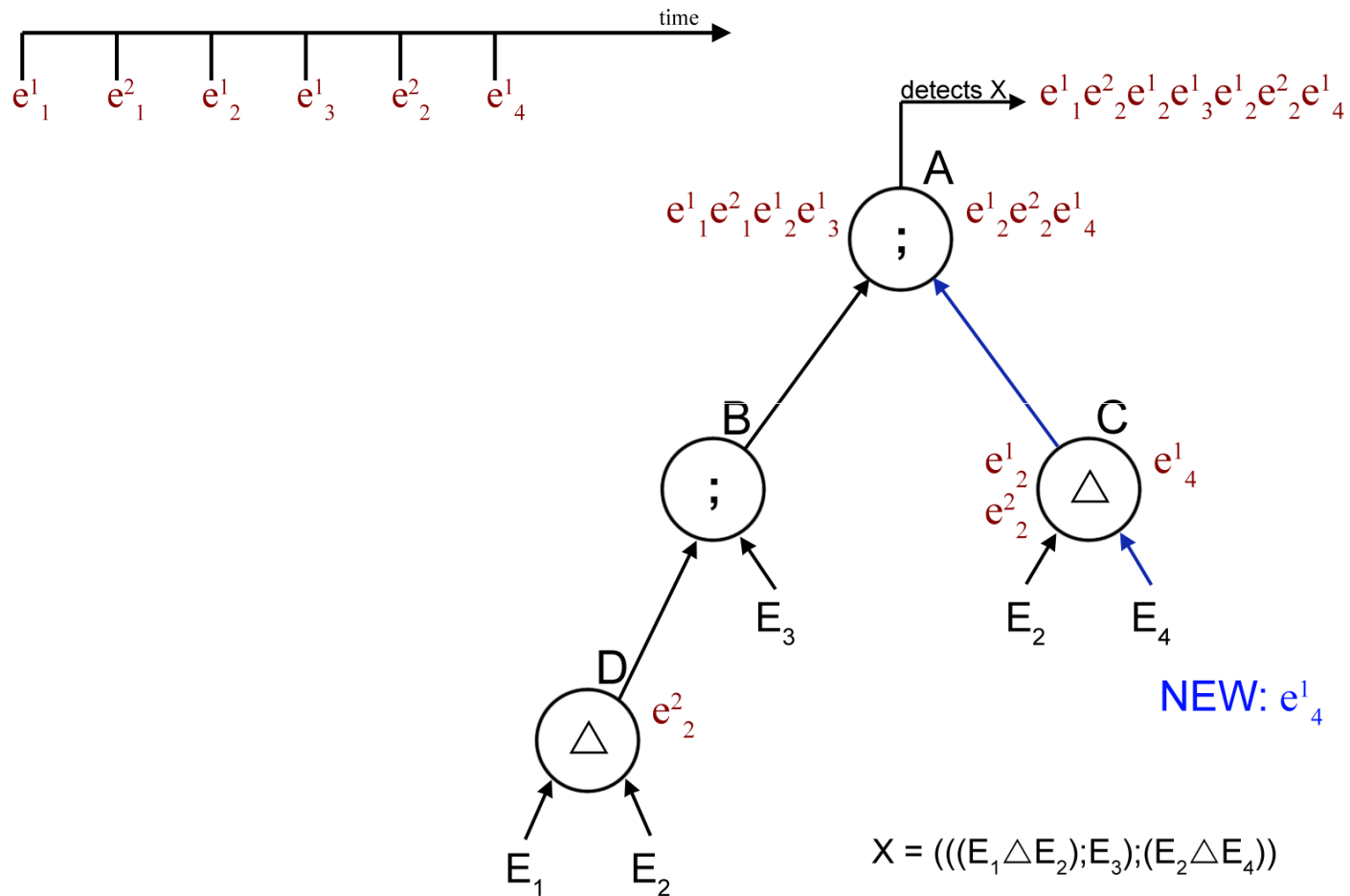
# CED: Chronicle Context



# CED: Continuous Context



# CED: Cumulative Context



# CED: Storage Requirements

- **Recent** - fixed size buffer
  - **Chronicle** - queue
  - **Continuous** - queue
  - **Cumulative** – much storage
- recent < cumulative < chronicle < continuous < unrestricted

# Active OODBMS Architecture

## Requirements:

### Events

- Primitive and Composite event detection
- Parameter computation
- Online and batch detection of composite events
- Inter-application (global) events

# Active OODBMS Architecture

- **Requirements**

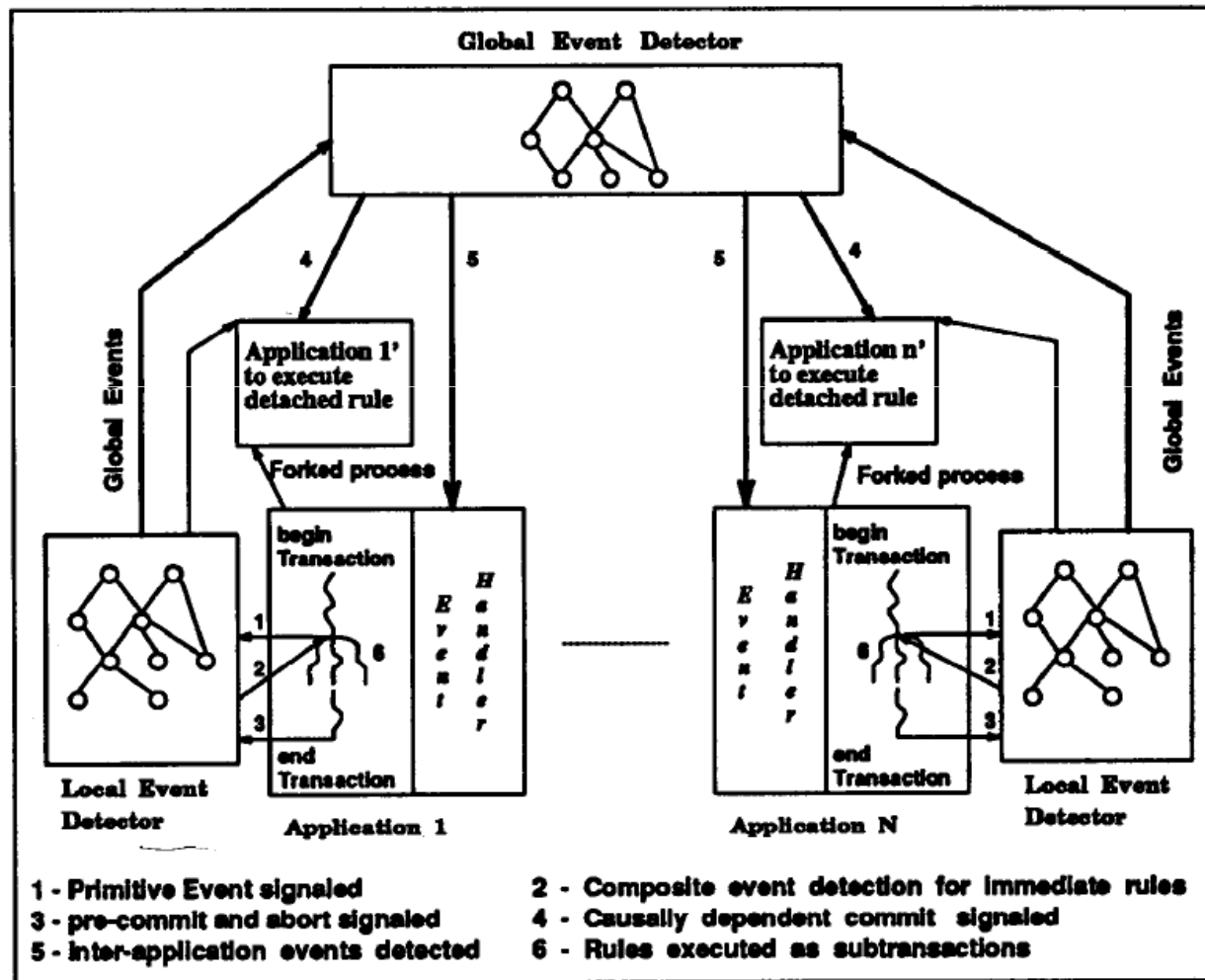
## Rules

- Multiple rules
- Nested rules
- Rule scheduling

# Sentinel Architecture

- Extends Open (passive) OODB system
- Threads for separating event detection from application execution

# Sentinel Architecture



# Conclusion

- Parameter contexts -> less storage overhead

# Composite Events for Active Databases

Thank you