



IBM Software Group – Event Processing technologies

Event Processing – Semantics

Part I: Setting the stage

Opher Etzion

A thin vertical black line extends downwards from the text 'Opher Etzion' to the top edge of the blue footer bar.

Semantics – of what ?

- ❑ Event Semantics
- ❑ Event Processing Network semantics
- ❑ The semantics of individual agent types

Event Semantics

- ❑ What do we need to know about events in general ?
- ❑ What do we need to know about an event relationships to other entities
- ❑ What do we need to know further about the event content

Event Processing Network Semantics

- ❑ What types of nodes ?
- ❑ What types of edges ?
- ❑ What is the general execution semantics?
- ❑ Is Context a semantic entity ?

Event processing – individual type of nodes

- Operational Semantics



IBM Software Group – Event Processing technologies

Event Processing – Semantics

Part II: Short answers

Opher Etzion

Event Semantics

□ Common properties:

- Source
- Time-point or interval
- Space coordinates or range
- Certainty factor

□ Relation to other entities:

- Reference and role (entities and through them reference data).
- Event Relations:
 - Causality – explicit / deduced (causality is implicit when event is derived somehow from another event)
 - Induced – generalization etc..

Event Processing semantics

□ Nodes:

- Producer
- Consumer
- Agent
- Channel – push / pull (periodic / on-demand).

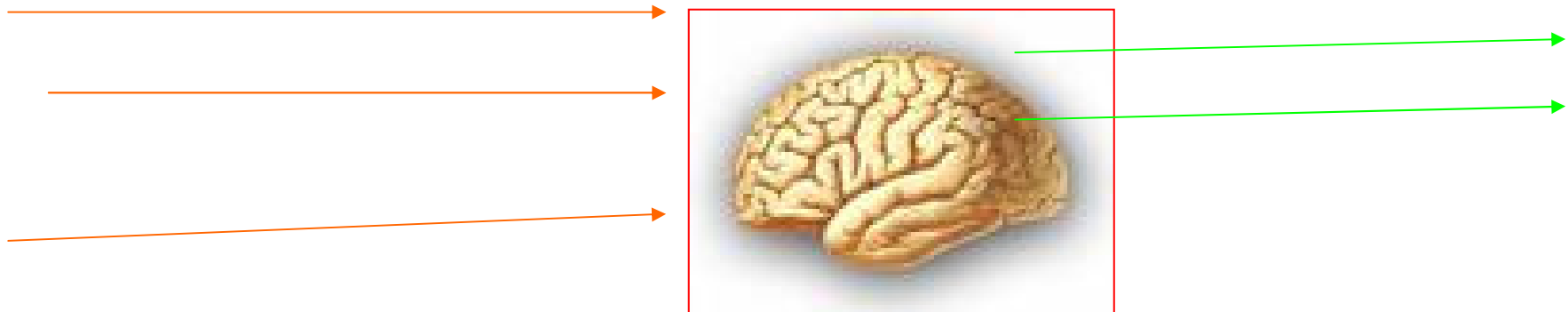
□ Edges:

- (P, CH); (CH, A); (A, CH); (CH, C); (CH, P); (C, CH)

Agents

An agent does three things:

- Selects which of the events in the cloud are relevant for it
- Perform the function
- Decide how to route the results



Input Cloud
= collection
of stream in
context

Scenario number 1: The heavy trading example

□ Given:

- **A stream of events of a single topic, about the activity in the stock market for a certain stock.**
- **An event is produced every 10 minutes when there is trade in the stock.**
- **Each event consists of: quote (current stock-quote), volume (an accumulated volume of traded events within these 10 minutes).**
- **A selection specification: "trigger an automatic trade program if the volume exceeds 300,000 3 times within an hour; pass as an argument the last quote and the sum of the 3 volume values".**

```
Select (Q3.quote, Q1.volume + Q2.volume + Q3.volume)
From Quote as Q1, Quote as Q2, Quote as Q3
Where (Q1.time < Q2.time and Q2.time < Q3.time and
Q1.volume > 300,000 and Q2.volume > 300,000 and Q3.volume
> 300,000)
Time-Interval: 1 hour
```

But now look at the actual stream

Event-Id	Time-Stamp	Quote	Volume
E1	9:00	33.23	
E2	9:10	33.04	320,000
E3	9:20	33.11	280,000
E4	9:30	33.01	400,000
E5	9:40	32.90	315,000
E6	9:50	33.04	320,000
E7	10:00	33.20	303,000
E8	10:10	33.33	219,000
E9	10:20	33.11	301,000
E10	10:40	33.00	210,000
E11	10:50	32.78	400,000
E12	11:00	32.70	176,000

- How many times the trade programming is triggered ;
- Which arguments are used in each triggering?

Scenario number 2: The matching example in continuous trading

□ Given:

- **A stream of events that belongs to 2 event-topics: buy-request and sell-request**
- **The buy-request consists of : stock, customer-id, quantity, upper-limit**
- **The sell-request consists of: stock, customer-id, quantity, lower-limit**
- **If a customer sends another sell or buy request before the previous one was settled, it overrides the previous one.**
- **The result is a "settlement" event that occurs which matches buy-request with sell-request such as $\text{upper-limit} \geq \text{lower-limit}$. The result event is: stock, buying-customer, selling-customer, minimum of quantities, lower-bound, upper-bound.**
- **The buy-request and sell-request do not participate in any matching again, even if not all desired quantity was settled.**
- **The matching should be "fair" in the sense of "first bid, first matched", where the time of a bid is considered as the time of the last override.**

Semantic Tuning decisions (some examples)

- ❑ **Decision 1: When is the selection applicable?**
- ❑ **Decision 2: Single or multiple time windows?**
- ❑ **Decision 3: When a detected event should be reported?**
- ❑ **Decision 4: Within an interval – one or many result streams?**
- ❑ **Decision 5. What quantifier is used in case of multiple instances?**
- ❑ **Decision 6: Can a consumed event be re-consumed?**
- ❑ **Decision 7: Are new events override previous events?**

Extended Language

Scenario 1

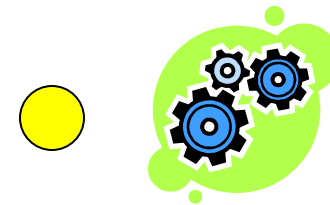
- Operator: Last 3 instances
- Select (Q (last).quote, sum (Q.volume))
- From Quote-instance as Q
- Where Q.volume > 300,000
- Within 1 hour
- Overlapping-intervals: no
- Detection-time : deferred
- Detection-type: single

Scenario 2

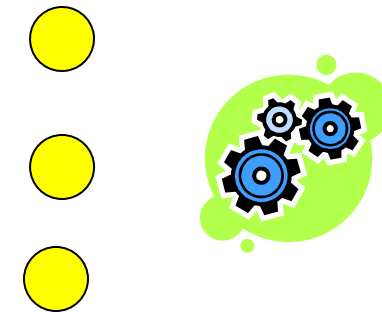
- Operator: And /* conjunction where the order does not matter */
- Select (buy-request.stock, sell-request.customer-id, buy-request.customer-id, Min (buy-request.quantity, sell-request.quantity), lower-bound, upper-bound)
- From relative first (buy-request, sell-request)
- Where (lower-bound <= upper-bound)
- Correlated-by: Stock
- Within-context: Continuous-trading
- Detection-time: Immediate
- Detection-type: Multiple
- Consumed-events: all
- Override: all (correlated-by Customer)

Event Processing Agents - Transformation

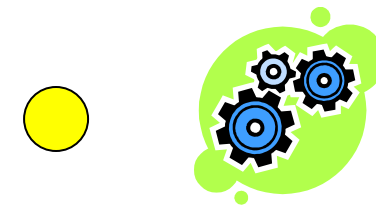
Translator: Both transformation and translation



Aggregator: statistical aggregator or concatenating events, can be stand alone agent or a scalar derivation. May be in network edge.

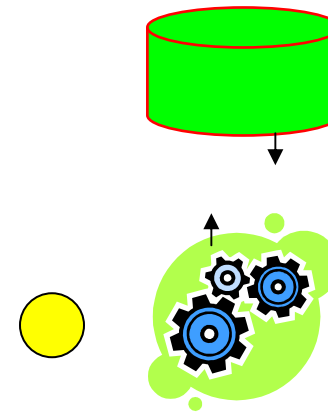


Splitter: Splits events to multiple events



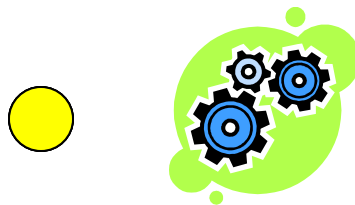
Event Processing Agents - Enrichment

Enricher: Enriches the content of events from reference data in databases, spreadsheets, Email messages, text files etc..



Event Processing Agents - Filtering

Filtering is a stateless agent that filters events. It is the most common agent, and a filtering agent has been required by many applications.



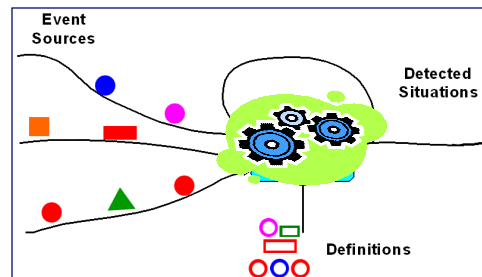
Event Processing Agents - Pattern Detection

Time series pattern detector:
"Set at a time" pattern detector.

Basic pattern detector: fixed set
of most common patterns.

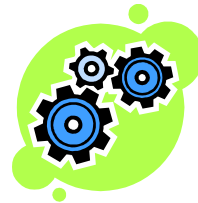
Advanced pattern detector:
various packaging of patterns,
pattern policies.

Output is "complex event"
representing the situation
detected by the specified
pattern

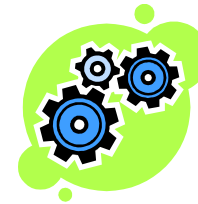


Event Processing Agents - Routing

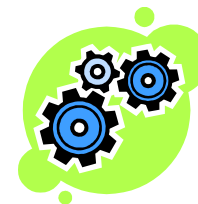
Itinerary-based routing



Subscription-based routing



Intelligent routing



Context

- ❑ First class citizen –
 - during working hours
 - For important customers
- ❑ Partition of the stream – each partition may have different rules
 - Partition by time
 - Partition by space
 - Partition by property (one or more)

Scenario number 1: The heavy trading example

□ Given:

- **A stream of events of a single topic, about the activity in the stock market for a certain stock.**
- **An event is produced every 10 minutes when there is trade in the stock.**
- **Each event consists of: quote (current stock-quote), volume (an accumulated volume of traded events within these 10 minutes).**
- **A selection specification: "trigger an automatic trade program if the volume exceeds 300,000 3 times within an hour; pass as an argument the last quote and the sum of the 3 volume values".**

```
Select (Q3.quote, Q1.volume + Q2.volume + Q3.volume)
From Quote as Q1, Quote as Q2, Quote as Q3
Where (Q1.time < Q2.time and Q2.time < Q3.time and
Q1.volume > 300,000 and Q2.volume > 300,000 and Q3.volume
> 300,000)
Time-Interval: 1 hour
```

But now look at the actual stream

Event-Id	Time-Stamp	Quote	Volume
E1	9:00	33.23	
E2	9:10	33.04	320,000
E3	9:20	33.11	280,000
E4	9:30	33.01	400,000
E5	9:40	32.90	315,000
E6	9:50	33.04	320,000
E7	10:00	33.20	303,000
E8	10:10	33.33	219,000
E9	10:20	33.11	301,000
E10	10:40	33.00	210,000
E11	10:50	32.78	400,000
E12	11:00	32.70	176,000

- How many times the trade programming is triggered ;
- Which arguments are used in each triggering?

Scenario number 2: The matching example in continuous trading

□ Given:

- **A stream of events that belongs to 2 event-topics: buy-request and sell-request**
- **The buy-request consists of : stock, customer-id, quantity, upper-limit**
- **The sell-request consists of: stock, customer-id, quantity, lower-limit**
- **If a customer sends another sell or buy request before the previous one was settled, it overrides the previous one.**
- **The result is a "settlement" event that occurs which matches buy-request with sell-request such as $\text{upper-limit} \geq \text{lower-limit}$. The result event is: stock, buying-customer, selling-customer, minimum of quantities, lower-bound, upper-bound.**
- **The buy-request and sell-request do not participate in any matching again, even if not all desired quantity was settled.**
- **The matching should be "fair" in the sense of "first bid, first matched", where the time of a bid is considered as the time of the last override.**

Semantic Tuning decisions (some examples)

- ❑ **Decision 1: When is the selection applicable?**
- ❑ **Decision 2: Single or multiple time windows?**
- ❑ **Decision 3: When a detected event should be reported?**
- ❑ **Decision 4: Within an interval – one or many result streams?**
- ❑ **Decision 5. What quantifier is used in case of multiple instances?**
- ❑ **Decision 6: Can a consumed event be re-consumed?**
- ❑ **Decision 7: Are new events override previous events?**

Extended Language

Scenario 1

- **Operator:** Last 3 instances
- **Select** (Q (last).quote, sum (Q.volume))
- **From** Quote-instance as Q
- **Where** Q.volume > 300,000
- **Within** 1 hour
- **Overlapping-intervals:** no
- **Detection-time :** deferred
- **Detection-type:** single

Scenario 2

- **Operator:** And /* conjunction where the order does not matter */
- **Select** (buy-request.stock, sell-request.customer-id, buy-request.customer-id, Min (buy-request.quantity, sell-request.quantity), lower-bound, upper-bound)
- **From** relative first (buy-request, sell-request)
- **Where** (lower-bound <= upper-bound)
- **Correlated-by:** Stock
- **Within-context:** Continuous-trading
- **Detection-time:** Immediate
- **Detection-type:** Multiple
- **Consumed-events:** all
- **Override:** all (correlated-by Customer)