

QONCEPT

Semantic Query Optimisation in Complex Event Processing Technologies

François Bry and Olga Poppe

November 19, 2009

Project Details in Brief

Project Acronym	QONCEPT
Action Line	Complex Event Processing, Databases, Knowledge presentation
Funding	DFG
Participants	LMU
Start Date	2009-11-01
End Date	2011-10-31 (2012-10-31)
Duration	2 years (3 years)
Related Projects	EMILI

Contents

1. Basic notions
 - Querying simple events
 - Complex Event Processing
 - CEP vs. query evaluation in DB
2. Query Optimisation in CEP Technologies
 - Query plan rewriting
 - Materialisation points
 - Event query constraints
 - Application-specific constraints
 - Relevance of events
3. Work plan

Querying Simple Events

Event = Message
(possible) Data format: Xcerpt data term

Use case: Online auction

```
auction {  
  auctionID [ 123 ],  
  category [ furniture ],  
  currency [ euro ],  
  sellerID [ 456 ]  
}
```

```
itemDescription {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  itemName [ sofa ],  
  value [ 1000 ]  
}
```

```
bid {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}
```

```
firstHammerBeat {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}
```

```
secondHammerBeat {  
  ...  
}  
thirdHammerBeat {  
  ...  
}
```

```
sell {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}
```

Querying Simple Events

Event = Message
(possible) Data format: Xcerpt data term
Peculiarity: Occurrence time interval $t=[b,e]$

```
auction {  
  auctionID [ 123 ],  
  category [ furniture ],  
  currency [ euro ],  
  sellerID [ 456 ]  
}  
t=[2,7]
```

```
itemDescription {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  itemName [ sofa ],  
  value [ 1000 ]  
}  
t=[2,3]
```

```
bid {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}  
t=[3,3]
```

```
firstHammerBeat {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}  
t=[4,4]
```

```
secondHammerBeat {  
  ...  
}  
thirdHammerBeat {  
  ...  
}
```

```
sell {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}  
t=[5,6]
```

Event stream = Set of events

Querying Simple Events

Event = Message
(possible) Data format: Xcerpt data term
Peculiarity: Occurrence time interval $t=[b,e]$

```
auction {  
  auctionID [ 123 ],  
  category [ furniture ],  
  currency [ euro ],  
  sellerID [ 456 ]  
}  
t=[2,7]
```

```
itemDescription {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  itemName [ sofa ],  
  value [ 1000 ]  
}  
t=[2,3]
```

```
bid {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}  
t=[3,3]
```

```
firstHammerBeat {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}  
t=[4,4]
```

```
secondHammerBeat {  
  ...  
}  
thirdHammerBeat {  
  ...  
}
```

```
sell {  
  auctionID [ 123 ],  
  itemID [ 789 ],  
  bidderID [ 007 ],  
  value [ 1001 ]  
}  
t=[5,6]
```

Event stream = Set of events

Querying Simple Events

Event query is formulated in an EQL, e.g. XChange

Tasks:

1. specifies a class of relevant events
2. extracts data from the events

ON

```
sell {{  
  var B → bidderID {{ }},  
  value [ var V ]  
}} where { var V > 10 000 }
```

DO

```
in { resource { 'file:bidder.xml' },  
  rich-bidders {{ insert var B }}  
}
```

END

Disadvantage: very limited

Querying Simple Events

Event query is formulated in an EQL, e.g. XChange

Tasks:

1. specifies a class of relevant events
2. extracts data from the events

ON

```
sell {{  
  var B → bidderID {{ }},  
  value [ var V ]  
}} where { var V > 10 000 }
```

DO

```
in { resource { 'file:bidder.xml' },  
  rich-bidders {{ insert var B }}  
}
```

END

Disadvantage: very limited

Complex Event Processing

- Complex event** = Message derived from certain combinations of events
- (possible) Data format: Xcerpt data term
- Peculiarity: Occurrence time interval $t=[b,e]$ encloses the time intervals of the constituent events
- Complex event query** = Set of simple event queries
is formulated in a CEQL, e.g. XChange^{EQ}

DETECT

```
all number-of-bids-per-item {  
  auctionID [ var A ],  
  var I,  
  number-of-bids [ count( all var B ) ]  
} group by { var A, var I }
```

ON

```
and {  
  event a: auction {{ auctionID [ var A ] }},  
  while a: collect var B → bid {{ auctionID [ var A ],  
    var I → itemID { { } } } }
```

```
}
```

END

Complex Event Processing

DETECT

```
uninteresting-items {  
  auctionID [ var A ], itemID [ var I ]  
}
```

ON

```
and {
```

```
  event a: itemDescription {{  
    auctionID [ var A ], itemID [ var I ], value [ var S ]  
  }},
```

```
  event b: sell {{  
    auctionID [ var A ], itemID [ var I ], value [ var E ]  
  }},
```

```
  event c: number-of-bids-per-item {{  
    auctionID [ var A ], itemID [ var I ], number-of-bids [ var N ]  
  }}
```

```
} where { var N ≤ 3, var E < var S +  $\frac{\text{var } S}{100}$ , a before b }
```

END

Complex Event Processing

Definition (CEP)

CEP denotes algorithmic methods for making sense of events by deriving higher-level knowledge (complex events) from lower-level events (simple events) in a timely fashion and permanently

CEP vs. Query Evaluation in DB

Similarities:

- the same data format, e.g. XML
- selecting, transforming, aggregating data

Differences:

	CEP	DB
Data	infinite, unpredictable	finite, available
Relationships between data items	not part of the data	part of the data
Time data	very important	not important
Evaluation	data-driven	query-driven
Optimisation	partial event history	indexing data

CEP vs. Query Evaluation in DB

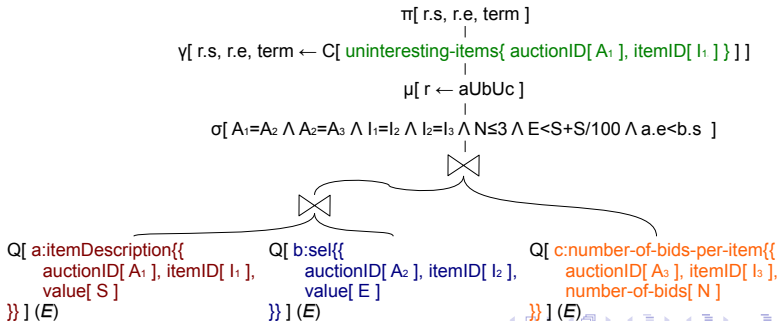
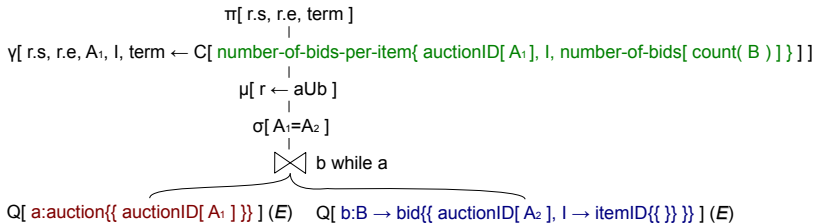
Similarities:

- the same data format, e.g. XML
- selecting, transforming, aggregating data

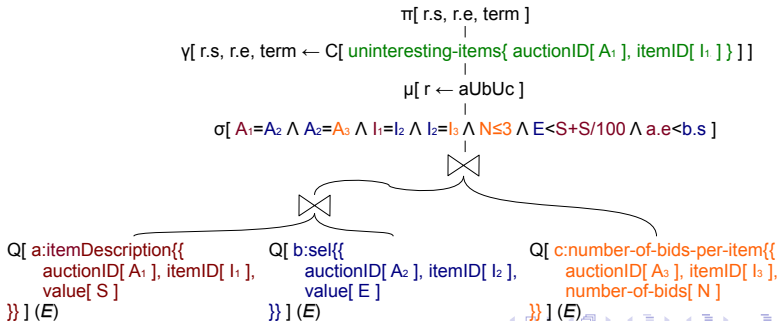
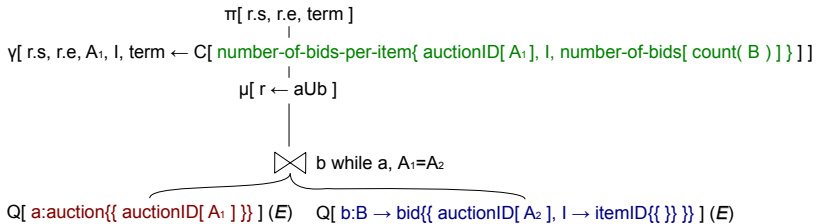
Differences:

	CEP	DB
Data	infinite, unpredictable	finite, available
Relationships between data items	not part of the data	part of the data
Time data	very important	not important
Evaluation	data-driven	query-driven
Optimisation	partial event history	indexing data

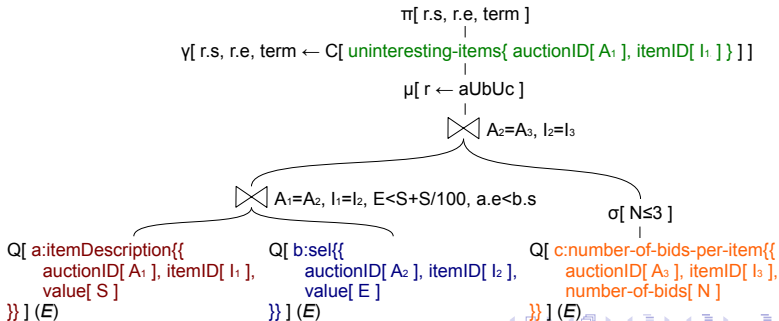
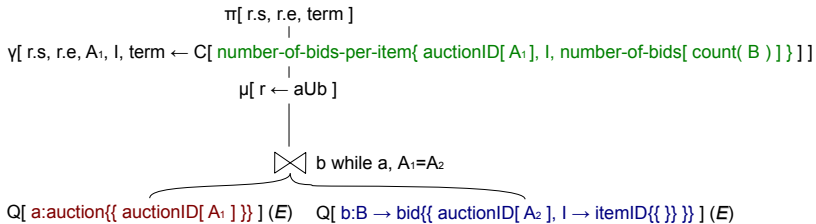
Query Plan



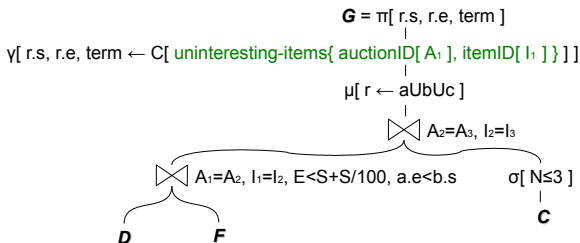
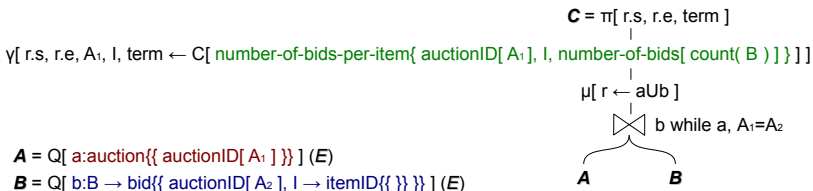
Query Plan Rewriting



Query Plan Rewriting



Materialisation Points

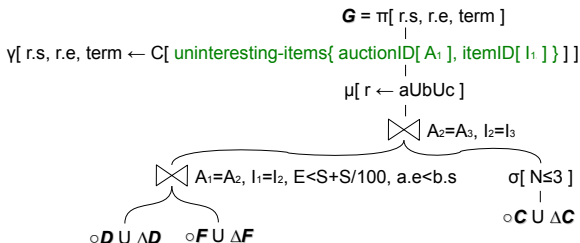
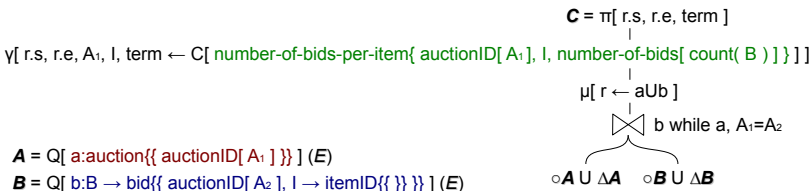


$D = Q[a:\text{itemDescription}\{ \{ \text{auctionID}[A_1], \text{itemID}[l_1], \text{value}[S] \} \}] (E)$

$F = Q[b:\text{sel}\{ \{ \text{auctionID}[A_2], \text{itemID}[l_2], \text{value}[E] \} \}] (E)$

$C = Q[c:\text{number-of-bids-per-item}\{ \{ \text{auctionID}[A_3], \text{itemID}[l_3], \text{number-of-bids}[N] \} \}] (E)$

Finite Differencing



$D = Q[a:\text{itemDescription}\{ \{ \text{auctionID}[A_1], \text{itemID}[l_1], \text{value}[S] \} \}] (E)$

$F = Q[b:\text{sel}\{ \{ \text{auctionID}[A_2], \text{itemID}[l_2], \text{value}[E] \} \}] (E)$

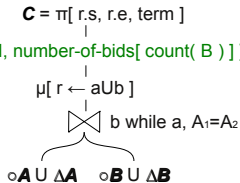
$C = Q[c:\text{number-of-bids-per-item}\{ \{ \text{auctionID}[A_3], \text{itemID}[l_3], \text{number-of-bids}[N] \} \}] (E)$

Finite Differencing

$\gamma[r.s, r.e, A_1, l, \text{term} \leftarrow C[\text{number-of-bids-per-item}\{ \text{auctionID}[A_1], l, \text{number-of-bids}[\text{count}(B)] \}]]$

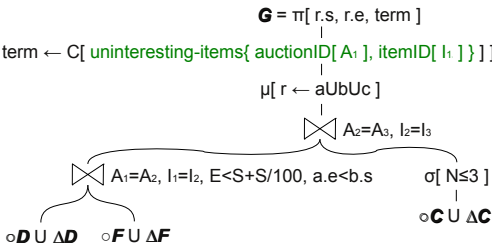
$A = Q[a:\text{auction}\{ \{ \text{auctionID}[A_1] \} \}] (E)$

$B = Q[b:B \rightarrow \text{bid}\{ \{ \text{auctionID}[A_2], l \rightarrow \text{itemID}\{ \} \} \}] (E)$



$\gamma[r.s, r.e, \text{term} \leftarrow C[\text{uninteresting-items}\{ \text{auctionID}[A_1], \text{itemID}[l_1] \}]]$

$\circ M = \circ M \cup \Delta M$

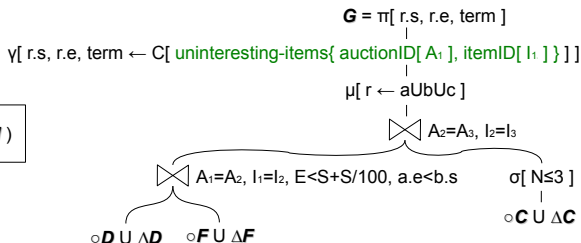
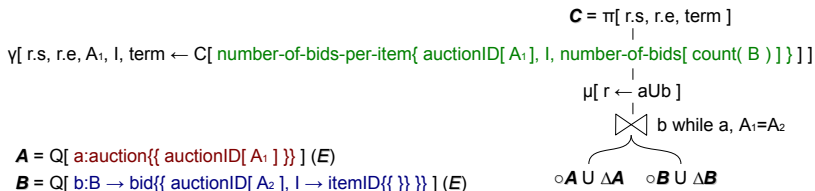


$D = Q[a:\text{itemDescription}\{ \{ \text{auctionID}[A_1], \text{itemID}[l_1], \text{value}[S] \} \}] (E)$

$F = Q[b:\text{sel}\{ \{ \text{auctionID}[A_2], \text{itemID}[l_2], \text{value}[E] \} \}] (E)$

$C = Q[c:\text{number-of-bids-per-item}\{ \{ \text{auctionID}[A_3], \text{itemID}[l_3], \text{number-of-bids}[N] \} \}] (E)$

Temporal Relevance of Events

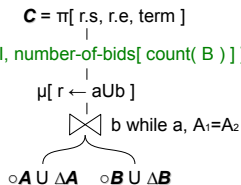


General Relevance of Events

$\gamma[r.s, r.e, A_1, l, \text{term} \leftarrow C[\text{number-of-bids-per-item}\{ \text{auctionID}[A_1], l, \text{number-of-bids}[\text{count}(B)] \}]]$

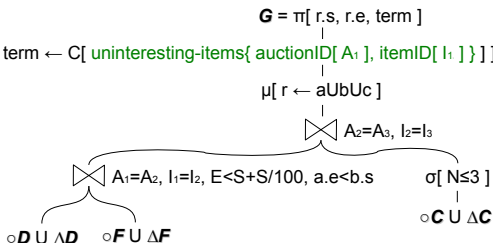
$A = Q[a:\text{auction}\{ \{ \text{auctionID}[A_1] \} \}] (E)$

$B = Q[b:B \rightarrow \text{bid}\{ \{ \text{auctionID}[A_2], l \rightarrow \text{itemID}\{ \} \} \}] (E)$



$\gamma[r.s, r.e, \text{term} \leftarrow C[\text{uninteresting-items}\{ \text{auctionID}[A_1], \text{itemID}[l_1] \}]]$

$\circ M = \sigma[GR_M] (\circ M U \Delta M)$



$D = Q[a:\text{itemDescription}\{ \{ \text{auctionID}[A_1], \text{itemID}[l_1], \text{value}[S] \} \}] (E)$

$F = Q[b:\text{sel}\{ \{ \text{auctionID}[A_2], \text{itemID}[l_2], \text{value}[E] \} \}] (E)$

$C = Q[c:\text{number-of-bids-per-item}\{ \{ \text{auctionID}[A_3], \text{itemID}[l_3], \text{number-of-bids}[N] \} \}] (E)$

Application Constraints

DETECT

```
all number-of-bids-per-item {  
  auctionID [ var A ],  
  var I,  
  number-of-bids [ count( all var B ) ]  
} group by { var A, var I }
```

ON

```
and {  
  event a: auction {{ auctionID [ var A ] }},  
  while a: collect var B → bid {{ auctionID [ var A ],  
    var I → itemID {{ } } } }
```

```
}  
END
```

⇒ The query returns a set of answers at the end of each auction

Application Constraints

DETECT

```
all number-of-bids-per-item {  
    auctionID [ var A ],  
    var I,  
    number-of-bids [ count( all var B ) ]  
} group by { var A, var I }
```

ON

```
and {  
    event a: auction {{ auctionID [ var A ] }},  
    while a: collect var B → bid {{ auctionID [ var A ],  
                                    var I → itemID {{ } } } }
```

```
}  
END
```

⇒ The query returns a set of answers at the end of each auction

Application Constraints

No bids (with *itemID* l_1) after the third hammer beat (with *itemID* l_1)

IF

and {

event t: thirdHammerBeat {{

auctionID [var A],

itemID [var l_1]

}},

event b: bid {{

auctionID [var A],

itemID [var l_2]

}}

} **where** { **b after t** }

THEN

var $l_1 \neq$ var l_2

END

⇒ The query returns an answer after each third hammer beat

Application Constraints

No bids (with *itemID* l_1) after the third hammer beat (with *itemID* l_1)

IF

and {

event t: thirdHammerBeat {{

auctionID [var A],

itemID [var l_1]

}},

event b: bid {{

auctionID [var A],

itemID [var l_2]

}}

} **where** { **b after t** }

THEN

var $l_1 \neq$ var l_2

END

⇒ The query returns an answer after each third hammer beat

Application Constraints

DETECT

```
uninteresting-items {  
  auctionID [ var A ], itemID [ var I ]  
}
```

ON

```
and {  
  event a: itemDescription {{  
    auctionID [ var A ], itemID [ var I ], value [ var S ]  
  }},  
  event b: sell {{  
    auctionID [ var A ], itemID [ var I ], value [ var E ]  
  }},  
  event c: number-of-bids-per-item {{  
    auctionID [ var A ], itemID [ var I ], number-of-bids [ var N ]  
  }}  
} where { var N ≤ 3, var E < var S +  $\frac{\text{var } S}{100}$  }
```

END

⇒ The query returns an answer after a *sell*-event for each item satisfying the conditions

Application Constraints

DETECT

```
uninteresting-items {  
  auctionID [ var A ], itemID [ var I ]  
}
```

ON

```
and {  
  event a: itemDescription {{  
    auctionID [ var A ], itemID [ var I ], value [ var S ]  
  }},  
  event b: sell {{  
    auctionID [ var A ], itemID [ var I ], value [ var E ]  
  }},  
  event c: number-of-bids-per-item {{  
    auctionID [ var A ], itemID [ var I ], number-of-bids [ var N ]  
  }}  
} where { var N ≤ 3, var E < var S +  $\frac{\text{var } S}{100}$  }
```

END

⇒ The query returns an answer after a *sell*-event for each item satisfying the conditions

Application Constraints

Price increases during the auction

IF

and {

event d: itemDescription {{

auctionID [var A], itemID [var I], value [var S]

}},

event b: bid {{

auctionID [var A], itemID [var I], value [var V]

}},

event s: sell {{

auctionID [var A], itemID [var I], value [var E]

}}

}

THEN

var E \geq var V, var V \geq var S

END

Application Constraints

IF

and {

event b_1 : bid {{

auctionID [var A], itemID [var I], value [var V_1]

}},

event b_2 : bid {{

auctionID [var A], itemID [var I], value [var V_2]

}}

} **where** { b_2 **after** b_1 }

THEN

var $V_2 >$ var V_1

END

\Rightarrow if var $V_{current} >$ var $S + \frac{var\ S}{100}$ then

var $E <$ var $S + \frac{var\ S}{100}$ is unsatisfiable for the respective item

Application Constraints

```
IF
  and {
    event  $b_1$ : bid {{
      auctionID [ var A ], itemID [ var I ], value [ var  $V_1$  ]
    }},
    event  $b_2$ : bid {{
      auctionID [ var A ], itemID [ var I ], value [ var  $V_2$  ]
    }}
  } where {  $b_2$  after  $b_1$  }
THEN
  var  $V_2 > \text{var } V_1$ 
END
```

\Rightarrow if var $V_{current} > \text{var } S + \frac{\text{var } S}{100}$ then
var $E < \text{var } S + \frac{\text{var } S}{100}$ is unsatisfiable for the respective item

Semantic Query Optimisation

Definition (Semantic Query Optimisation)

In DB the idea of using *application-specific knowledge* about (possible) data to make query evaluation more efficient, is called semantic query optimisation

Constraint Language

DETECT

```
uninteresting-items-bought-by-007 {  
  auctionID [ var A ], itemID [ var I ]  
}
```

ON

```
and {
```

```
  event a: itemDescription {{  
    auctionID [ var A ], itemID [ var I ], value [ var S ]  
  }},
```

```
  event b: sell {{  
    auctionID [ var A ], itemID [ var I ], bidderID[ 007 ], value [ var E ]  
  }},
```

```
  event c: number-of-bids-per-item {{  
    auctionID [ var A ], itemID [ var I ], number-of-bids [ var N ]  
  }}
```

```
} where { var N ≤ 3, var E < var S +  $\frac{\text{var } S}{100}$  }
```

END

Constraint Language

007 may buy not more than 5 items per auction

IF

```
and {  
  event a: auction {{  
    auctionID [ var A ]  
  }},  
  while a: collect var S → sell {{  
    auctionID [ var A ], bidderID [ 007 ]  
  }}  
}
```

THEN

```
count( all var S ) ≤ 5
```

END

⇒ After the 5th *sell*-event (with *bidderID* 007) in each auction
the query is unsatisfiable till the end of the auction

Constraint Language

007 may buy not more than 5 items per auction

IF

and {

event **a**: auction {{
 auctionID [var A]

}},

while **a**: **collect** var S \rightarrow sell {{
 auctionID [var A], bidderID [007]

}}

}

THEN

count(**all** var S) \leq 5

END

\Rightarrow After the 5th *sell*-event (with *bidderID* 007) in each auction
the query is unsatisfiable till the end of the auction

Constraint Language

007 may not buy

```
IF
  sell {{ bidderID [ var B ] }}
THEN
  var B  $\neq$  007
END
```

⇒ The query is unsatisfiable

Constraint Language

007 may not buy

```
IF
    sell {{ bidderID [ var B ] }}
THEN
    var B  $\neq$  007
END
```

\Rightarrow The query is unsatisfiable

Constraint Language

1. Based on XChange^{EQ}
2. User-friendly
3. Expressive
4. Efficient

Relevance of Events

1. Automatically derived from
 - Event query constraints
 - Application-specific constraints
2. Dynamic
3. Manifold
4. Minimal
5. Efficient

Work Plan

1. Survey of EQLs
2. Constraint language
3. Query planner should:
 - link the constraint language with XChange^{EQ},
 - suggest many query plans and
 - choose the best one
4. Experimental analysis

Main goal: Combined usage of different semantic query optimisation techniques in CEP

Work Plan

1. Survey of EQLs
2. Constraint language
3. Query planner should:
 - link the constraint language with XChange^{EQ},
 - suggest many query plans and
 - choose the best one
4. Experimental analysis

Main goal: Combined usage of different semantic query optimisation techniques in CEP