

QONCEPT:

Semantic Query Optimisation in CEP Technologies

Olga Poppe

Supervisor: François Bry

Institute for Informatics, LMU Munich, Germany

July 8, 2011

- 1 Semantic Query Optimisation
- 2 Metadata
 - ESCL: Event Stream Constraint Language
 - IHTA: Instantiating Hierarchical Timed Automata
- 3 Method
- 4 Conclusion and Future Work

- 1 Semantic Query Optimisation
- Metadata
 - ESCL: Event Stream Constraint Language
 - IHTA: Instantiating Hierarchical Timed Automata
- Method
- Conclusion and Future Work

SQO is the use of metadata for query optimisation such as:

- 1 Result by contradiction
- 2 Result by transformation
- 3 Join elimination
- 4 Join introduction
- 5 Predicate elimination
- 6 Predicate introduction

SQO is the use of metadata for query optimisation

Origins of application semantics:

- 1 Application logic
 - US visa application
 - Auction
 - Trip booking
 - Product order
- 2 Physical laws
 - Sensor network

Metadata used in database systems:

- Relation schema + Constraints

Peculiarities of SQO of CEP compared to SQO of DB Systems

1 Time dependency

⇒ Event Stream Constraint Language

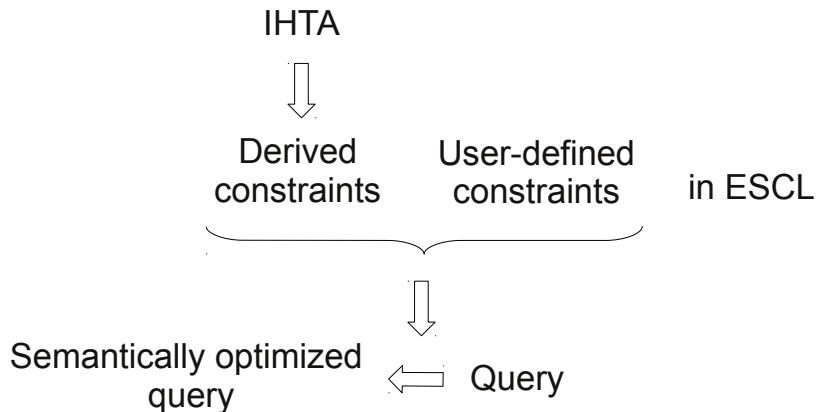
⇒ Instantiating Hierarchical Timed Automata

2 Standing queries moving data

⇒ Multi-query optimisation is important

⇒ Constraint propagation

⇒ Static SO may be expensive



- 1 Semantic Query Optimisation
- 2 Metadata
 - ESCL: Event Stream Constraint Language
 - IHTA: Instantiating Hierarchical Timed Automata
- 3 Method
- 4 Conclusion and Future Work

Kinds of constraints in CEP applications:

- Cardinality
- Functional dependencies:
 - Causality
 - Temporal relations
 - Spatial relations
 - Data relations

Simple but expressive constraints for CEP by means of:

- Event and state queries
- Identification of matched events and states
- Quantification

Constraints languages:

- DDL [14],
- DTD [25, 24], [18, 19], [9], [16], [15], [21],
- Regular expressions [12], [11], [13], [26],
- Denial rules [10],
- Language independent formulas [25, 24], [9], [13], [6], [4], [27], [3, 22].

Particular kind(s) of constraints:

- Cardinality constraints [25, 24], [9], [14], [4], [27], [3, 22],
- Causal dependencies [25, 24], [10],
- Conditions on event data [12], [11], [4], [26],
- Temporal dependencies [25, 24], [9], [12], [11], [13], [14], [6], [4], [26], [10].

ESCL \approx Denial rules

- + Identifiers
- + Temporal and other relations
- + Quantifiers
- + Modules
- + Local definitions

There is exactly one item description for each item presented in an auction.

```
WHILE state: auction( auctionID(A) )
LET
  IF event i: itemDescription( auctionID(A), itemID(I) )
  THEN  $\exists_{=1} i$ 
  END
END
```

Bidder with ID 007 may not participate in any auction because of his bad behaviour.

```
WHILE state: auction( auctionID(A) )
LET
  IF event: bid( auctionID(A), itemID(I), bidderID(B) )
  THEN var B  $\neq$  007
  END
END
```

Examples of ESCL Constraints

Price for an item increases during an auction.

```

WHILE
  state: itemOffer( itemID(I) )
LET
  LET
    DETECT
      event: e( itemID(I), value(V1) )
    ON or {
      event: bid( itemID(I), value(V1) ),
      event: hammerBeat( itemID(I), value(V1) )
    }
  END
  IN
    IF and { event: itemDescription( itemID(I), value(V2) ),
             event: e( itemID(I), value(V1) ) }
    THEN var V2 ≤ var V1
    END
    IF and { event i: e( itemID(I), value(V1) ),
             event b: bid( itemID(I), value(V2) ),
             i before b }
    THEN var V1 < var V2
    END
    IF and { event i: e( itemID(I), value(V1) ),
             event h: hammerBeat( itemID(I), value(V2) ),
             i before h }
    THEN var V1 ≤ var V2
    END
    IF and { event: e( itemID(I), value(V1) ),
             event: sell( itemID(I), value(V2) ) }
    THEN var V1 ≤ var V2
    END
  END
END

```

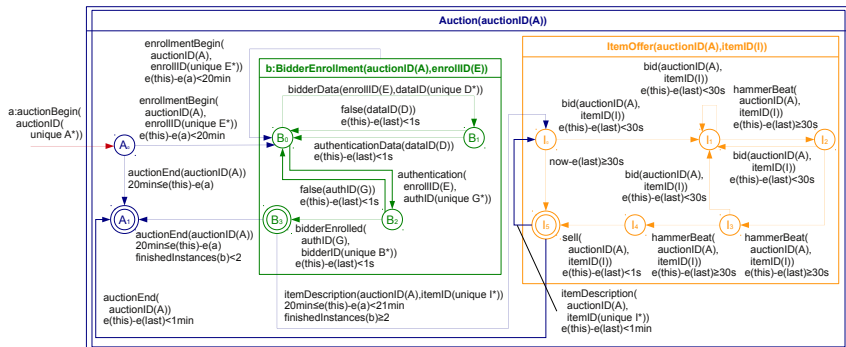
... without:

- Disjunctions in the body
- Local definitions
- Modules

```
 $\exists=1$  i ←  
  event i: itemDescription( auctionID(A), itemID(I) ),  
  state s: auction( auctionID(A) ),  
  i during s  
  
var B ≠ 007 ←  
  event b: bid( auctionID(A), itemID(I), bidderID(B) ),  
  state s: auction( auctionID(A) ),  
  b during s
```

- Set of constraints
- UML diagrams
- Petri nets
- Flowcharts
- Automata

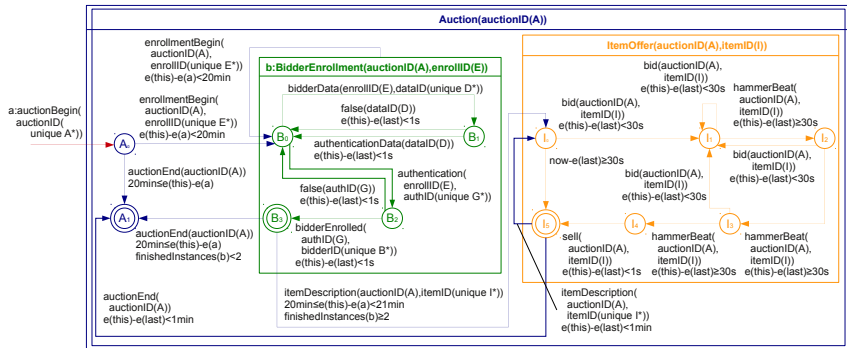
IHTA: Instantiating Hierarchical Timed Automata



IHTA vs. Nondeterministic Finite Automata [23]:

- Atomic and non-atomic states
- Transitions

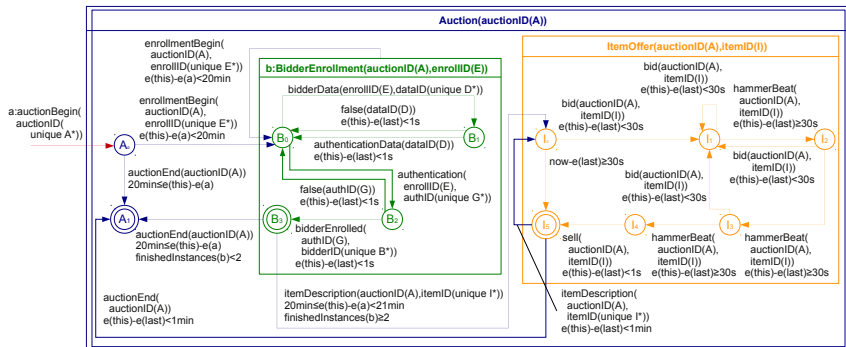
IHTA: Instantiating Hierarchical Timed Automata



IHTA vs. Timed Automata [2], [5], [1], [20]:

- Events and states carry data
- Temporal constraints on the beginning and the end of the occurrence time of matched events

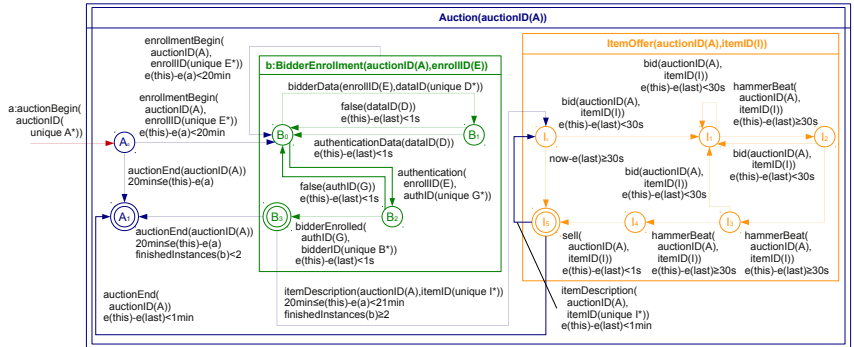
IHTA: Instantiating Hierarchical Timed Automata



IHTA vs. Hierarchical Timed Automata [8], Statecharts [17]:

- Modular, means for abstraction, extensible
- Arbitrary number of concurrent processes

Constraints Expressed by IHTA



- 1 Cardinality
- 2 Functional dependencies
 - Temporal relations
 - Causal relations
 - Relations between event data

all automata

all timed automata

all automata

IHTA

- 1 Semantic Query Optimisation
- 2 Metadata
 - ESCL: Event Stream Constraint Language
 - IHTA: Instantiating Hierarchical Timed Automata
- 3 Method
- 4 Conclusion and Future Work

Set of queries of the form $H_Q \leftarrow B_Q$, e.g.:

```
bidNumber( auctionID(A), itemID(I), number(count(all A,I), bidderID(007) ) ←
  event i: itemDescription( auctionID(A), itemID(I) ),
  event b: bid( auctionID(A), itemID(I), bidderID(007) ),
  state s: auction( auctionID(A) ),
  i during s,
  b during s
```

Set of constraints of the form $H_C \leftarrow B_C$, e.g.:

```
var B ≠ 007 ←
  event b: bid( auctionID(A), itemID(I), bidderID(B) ),
  state s: auction( auctionID(A) ),
  b during s
```

Let S_Q be set of queries and
 S_C be set of constraints

```
FOR EACH  $H_Q \leftarrow B_Q \in S_Q$ 
  FOR EACH  $H_C \leftarrow B_C \in S_C$ 
    IF  $B_C$  subsums  $B_Q$  with  $\sigma$ 
      THEN rewrite  $B_Q$  to  $B_Q, H_C \sigma$ 
    END
  END
END
```

Return S_Q

Let S_Q be set of queries and
 S_C be set of constraints

```
FOR EACH  $H_Q \leftarrow B_Q \in S_Q$ 
  FOR EACH  $H_C \leftarrow B_C \in S_C$ 
    IF  $B_C$  subsums  $B_Q$  with  $\sigma$  and
      a SO heuristic is applicable to  $H_Q \leftarrow B_Q$  and  $H_C \sigma$ 
    THEN rewrite  $B_Q$  to  $B_Q, H_C \sigma$ 
      apply SO heuristics to  $B_Q$ 
    END
  END
END
END

Return  $S_Q$ 
```

Residue method [7]

- ① Result by contradiction
- ② Result by transformation
- ③ Join elimination
- ④ Join introduction
- ⑤ Predicate elimination
- ⑥ Predicate introduction

Query:

```
bidNumber( auctionID(A), itemID(I), number(count(all A,I)), bidderID(007) ) ←
  event i: itemDescription( auctionID(A), itemID(I) ),
  event b: bid( auctionID(A), itemID(I), bidderID(007) ),
  state s: auction( auctionID(A) ),
  i during s,
  b during s
```

Constraint:

```
var B ≠ 007 ←
  event b: bid( auctionID(A), itemID(I), bidderID(B) ),
  state s: auction( auctionID(A) ),
  b during s
```

- 1 Result by contradiction
- 2 Result by transformation
- 3 Join elimination
- 4 Join introduction
- 5 Predicate elimination
- 6 Predicate introduction

Query:

```
bidNumber( auctionID(A), itemID(I), number(count(all A,I)), bidderID(007) ) ←
  event i: itemDescription( auctionID(A), itemID(I) ),
  event b: bid( auctionID(A), itemID(I), bidderID(007) ),
  state s: auction( auctionID(A) ),
  i during s,
  b during s
{ 007 ≠ 007 }
```

Constraint:

```
var B ≠ 007 ←
  event b: bid( auctionID(A), itemID(I), bidderID(B) ),
  state s: auction( auctionID(A) ),
  b during s
```

Let Q be a query,
let Q' be a semantically optimised query Q using a constraint set C .
 Q and Q' must be *semantically equivalent*, i.e.
return the same results with respect to all streams satisfying C .

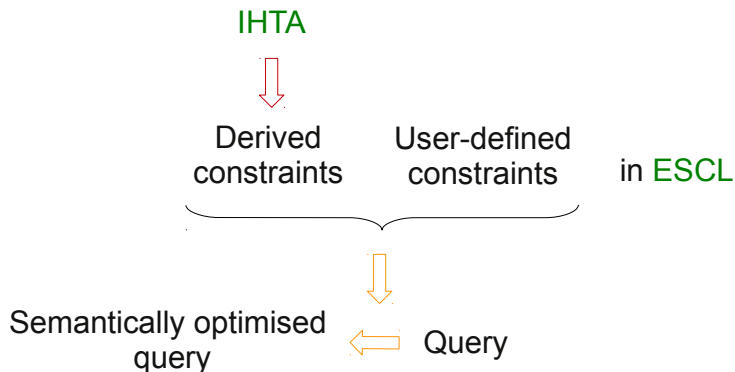
The SO algorithm is proven to

- terminate,
- be correct,
- be in $O(Q \cdot C \cdot S \cdot H)$ where
 - Q is the number of queries,
 - C is the number of constraints,
 - S is the costs for subsumption of CEP queries,
 - H is the costs for the application of SO heuristics

Dimensions:

- 1 Event and state literals
- 2 Temporal and other relations
- 3 Complex events / query propagation
- 4 Cardinality
- 5 Constraints

- 1 Semantic Query Optimisation
- 2 Metadata
 - ESCL: Event Stream Constraint Language
 - IHTA: Instantiating Hierarchical Timed Automata
- 3 Method
- 4 Conclusion and Future Work



- Derivation of constraints from IHTA
- Subsumption of CEP queries
- Query planner, cost models, experimental evaluation
- Constraint propagation
- Related work

- [1] Y. Abdeddaim, E. Asarin, and O. Maler.
Scheduling with Timed Automata.
Elsevier Science, 2006.
- [2] R. Alur and D. L. Dill.
Automata for modeling real-time systems.
In *Proc. Int. Conf. on Automata, Languages and Programming*, 1990.
- [3] R. Avnur and J. M. Hellerstein.
Eddies: Continuously adaptive query processing.
In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2000.
- [4] S. Babu, U. Srivastava, and J. Widom.
Exploiting k-constraints to reduce memory overhead in continuous queries over data streams.
ACM Transactions on Database Systems, 29(3), 2004.
- [5] J. Bengtsson and W. Yi.
Timed automata: Semantics, algorithms and tools.
In *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2004.
- [6] F. Bry and M. Eckert.
Temporal order optimizations of incremental joins for composite event detection.
In *Proc. Int. Conf. on Distributed Event-Based Systems*. ACM, 2007.
- [7] U. S. Chakravarthy, J. Grant, and J. Minker.
Logic-based approach to semantic query optimization.
volume 15, pages 162–207. ACM, 1990.

- [8] A. David and M. D. Mölle.
From HUPPAAL to UPPAAL - a translation from Hierarchical Timed Automata to Flat Timed Automata, 2001.
- [9] Y. Diao and M. Franklin.
Query processing for high-volume XML message brokering.
In *Proc. Int. Conf. on Very Large Data Bases*, pages 261–272. VLDB Endowment, 2003.
- [10] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W.-P. Hsiung, and K. S. Candan.
Runtime semantic query optimization for event stream processing.
In *Proc. Int. Conf. on Data Engineering*, pages 676–685. IEEE Computer Society, 2008.
- [11] L. Ding, N. Mehta, E. A. Rundensteiner, and G. T. Heineman.
Joining punctuated streams.
In *Proc. Int. Conf. on Extending Database Technology*, 2004.
- [12] L. Ding, E. A. Rundensteiner, and G. T. Heineman.
MJoin: A metadata-aware stream join operator.
In *Proc. Int. Workshop on Distributed Event-Based Systems*, pages 1–8. ACM, 2003.
- [13] P. M. Fischer, K. S. Esmaili, and R. J. Miller.
Stream schema: Providing and exploiting static metadata for data stream processing.
In *Proc. Int. Conf. on Extending Database Technology*, 2010.
- [14] L. Golab, T. Johnson, N. Koudas, D. Srivastava, and D. Toman.
Optimizing away joins on data streams.
In *Proc. Int. Workshop on Scalable Stream Processing System*, pages 48–57. ACM, 2008.
- [15] T. J. Green, G. Miklau, M. Onizuka, and D. Suciu.
Processing XML streams with deterministic automata.
In *Proc. Int. Conf. on Database Theory*, 2002.

- [16] A. K. Gupta and D. Suciú.
Stream processing of XPath queries with predicates.
In *Proc. Int. Conf. on Management of Data*, 2003.
- [17] D. Harel.
Statecharts: A visual formalism for complex systems.
volume 8. Elsevier Science, 1987.
- [18] C. Koch, S. Scherzinger, N. Schweikardt, and B. Stegmaier.
FluXQuery: An optimizing XQuery processor for streaming XML data.
In *Proc. Int. Conf. on Very Large Data Bases*, 2004.
- [19] C. Koch, S. Scherzinger, N. Schweikardt, and B. Stegmaier.
Schema-based scheduling of event processors and buffer minimization for queries on structured data streams.
In *Proc. Int. Conf. on Very Large Data Bases*, 2004.
- [20] S. Lasota and I. Walukiewicz.
Alternating Timed Automata.
volume 9. ACM, 2008.
- [21] B. Ludäscher, P. Mukhopadhyay, and Y. Papakonstantinou.
A transducer-based XML query processor.
In *Proc. Int. Conf. on Very Large Data Bases*, pages 227–238. VLDB Endowment, 2002.
- [22] S. Madden, M. Shah, J. M. Hellerstein, and V. Raman.
Continuously adaptive continuous queries over streams.
In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 49–60. ACM, 2002.

- [23] U. Schöning.
Theoretische Informatik - kurzgefasst.
Spektrum Akademischer Verlag, 2003.
- [24] H. Su, E. A. Rundensteiner, and M. Mani.
Semantic query optimization in an automata-algebra combined XQuery engine over XML streams.
In *Proc. Int. Conf. on Very Large Data Bases*, 2004.
- [25] H. Su, E. A. Rundensteiner, and M. Mani.
Semantic query optimization for XQuery over XML streams.
In *Proc. Int. Conf. on Very Large Data Bases*, 2005.
- [26] P. A. Tucker, D. Maier, T. Sheard, and P. Stephens.
Using production schemas to characterize strategies for querying over data streams.
IEEE Transactions on Knowledge and Data Engineering, 19(9), 2007.
- [27] S. D. Viglas and J. F. Naughton.
Rate-based query optimization for streaming information sources.
In *Proc. Int. Conf. on Management of Data*, 2002.