Complex Event Processing

- Complex Event Processing (CommonSens)
- Adaptive windows in DSMS (health monitoring)

Vera Goebel
Today’s Computing Paradigm

Device centric I/O

Input → Computing Device → Output

Human interaction, respectively human in the loop
Future Networked Computing

From Human Computer Interaction (HCI) to Computer Environment Interaction (CEI)

Networked computing devices, human interaction
CEP: Example Applications

• Algorithmic stock trading
• Credit card fraud detection
• Business activity monitoring
• Security monitoring
• Health care

• DSMS <-> CEP
Health Care Applications

• Integrated patient monitoring
• Telemonitoring of human physiological data
• Tracking and monitoring doctors and patients inside a hospital
• Tracking and monitoring patients and rescue personnel during rescue operations
Motivation I

- **DSMS technology**
  - A data stream of tuples
    - A tuple consists of attributes
      - Values
      - Timestamps
    - In-memory real-time processing
  - Perform projections and aggregations
    - *Give me all tuples from company X*
    - *Give me the average packet size in the network from the last five minutes*
    - *Tell me if the temperature is lower than 15 degrees*
  - Uses windowing for blocking operators
  - Based on SQL
Motivation II

• Beyond projections and aggregation: We want to know *more* about the data stream!
  • When does a specific pattern occur?
    • *Tell me when the stock prize from company X increases by 10% for ten minutes before decreasing*
  • Order of consecutive events
    • *Only tell me when the goods have been taken out of the shop without being purchased in the counter*
    • *How long does it take from a person falls until he gets help?*
  • Concurrency
    • *Tell me when the cooker is on while the person has left the home*
Events … (Terminology)

- **Event**: change of state
- **Event Processing**: method of tracking and analyzing (processing) of streams of information (data) about things that happen (events).
- **Complex Event Processing (CEP)**: event processing that combines data from multiple sources to infer events or patterns for complicated situations.
Automated Home Care

• Increasing ratio of elders
  – Need alternative approaches
  – Safety

• Use sensors
  – Placed in the home
    • Motion
    • Heart beat
    • ...
  – Heterogeneous

• Monitored person
  – Lives in the home
Problem

• Application programmer
  • Instructs the home care system
  • Programs at a higher level

• Need for a system for automated home care
  • Use complex event processing (CEP)
  • Simplify the work for the application programmer
  • Assisting personalisation
    • Current home, sensors and monitored person
**Event Model**

- **Event**: An interesting state or state transition
  - Atomic event
    - State value
    - Two timestamps
      - Define duration of event
    - Location of interest (LoI)
      - Define location of event
  - Complex event
    - A set of atomic events which occur concurrently or consecutively
Challenges & Requirements

• How to integrate the different sensors?
• How to model their abilities … (semantics)
• Application developers, e.g., home care:
  • Domain knowledge
  • Different environments
  • Different installations

• Requirements:
  • Support for placement and coverage area calculation
  • Re-use & easy personalization
  • Avoid to address particular sensors
    • Capabilities
    • Locations-of-Interests (LoI)
Environment Model

• Important properties in the home
  – Sensor placement

• Objects
  – Rooms, walls, furniture, monitored person
  – Shapes
    • 3D coordinates
    • Describe boundaries of objects
  – Permeability
    • How the object affects sensor signals

• Environment/home
  – A set of objects with shapes and permeability values
  – Contains user defined locations of interest (i.e., shapes)
Sensor Model

- **Capabilities**
  - The type of state variables a sensor can observe

- **Physical sensors**
  - Convert analogue signals to data tuples
  - Provide a set of capabilities
  - Cover objects or areas in the environment
  - Should cover locations of interest (LoIs) in the environment
Sensor Model (cont.)

• **External sources**
  – Provide a set of capabilities
  – Return stored/historical data tuples
    • DBMSs
    • Haar classifiers for face recognition
    • ...

• **Logical sensors**
  – Deliver aggregates of data tuples from other sensors
  – Provide a set of capabilities
  – Depend on a set of capabilities
Sensor Model (cont.)

- Multimodality
  - Different sensors can provide the same capabilities
  - One sensor can provide several capabilities

- TakingMedication

  Logical sensor

  ContinuousPictureMatrix

  TakingMedicationPattern

- Physical sensor

  Logical sensor

  SinglePictureMatrix

- External source: Haar

  Camera

  FaceRecognition

  MedicineBottleRecognition

  External source: Haar
Query Language

- Describes complex events
- **Atomic query**
  - **Condition**
    - Capability
    - Operator
    - Value
  - **LoI**
  - **Temporal properties**
- **Complex query**
  - List of atomic queries
  - Operators
    - `&&`, `||`, `!`
  - **Relation**
    - `->`
  - **Concurrency classes**
    - `equals`, `starts`, `finishes`, `during`, `overlaps`
CommonSens Life Cycle Phases

1. Integrate model instances
   1. Model environment
   2. Write queries
   3. Find available sensors
      • Provide correct capabilities

2. **Place sensors in the environment**
   1. Through simulation
   2. In the real environment

3. **Instantiate queries**

4. Perform complex event processing
Core Idea

Sensor Model:
- Physical & logical
- Capabilities
- Signal Types
- Coverage area

Event Model:
- Atomic
- Complex
- Temporal
- Spatial

Environment Model:
- Objects
- Permeability
- Sensors

Statements (Lols & Capabilities)

Sensor placement

Statement instantiation

State machine creation

Data gathering

Evaluation

A priori

Runtime, i.e., event processing
Sensor Placement

• Use signal propagation models
• Permeability values of objects
  – Radio signals go through walls
  – Light is stopped by walls
• We assume that the data we obtain is correct
• Signals modelled as rays
Sensor Placement (cont.)

• Approximating Lols for query instantiation:

Isec(loi)  NoIsec(loi)  LoIApprox(loi)  FPProb(loi)
Deviation Detection

• A considerable number of things can go wrong
  – Do not get up in the morning
  – Temperature too high
  – Leaves the home during the night

• Hard for application programmer to identify all this

• Turn problem upside down
  – Only query expected behaviour
    • Breakfast is at 8
    • Temperature = 20 degrees
    • Should be home all night
Deviation Detection (cont.)

• However, query processor does not report anything if there is no match
  • Not enough to negate query
    • Complex queries consist of many atomic queries
    • Evaluation of the query will continue if conditions are not matched

• Important to report immediately when the conditions are not met
Deviation Detection (cont.)

All sequences of data tuples related to all queries

The sequences that are related to one particular query

The sequences that match the particular query

$$D = N \setminus E$$
State of the Art

- Events
  - Many different interpretations and models
- Sensor technology
  - Comprehensive models
- Sensor placement
  - Office environments
  - Multimodal surveillance
- Query languages
  - No support for deviations
  - Lack of concurrency except ‘AND’
- Personalisation
  - Not much discussed within the domain
  - Not combined with CEP yet
- Deviation detection
  - Use statistical methods

CommonSens
Integrates models and concepts related to multimodal complex event processing in automated home care
A Declarative Approach

• Just as in DSMSs
  • Describe what should happen
  • Perform aggregations on the data
• Describe what we want, not *how* to get it
• SQL-like syntax is common
  SELECT attributes
  FROM event streams
  WHERE attribute values…

• In addition, the systems have syntax for describing patterns
  • E.g. in Esper

© Jarle Søberg, 2010
CEP Systems

• Research:
  • YFilter++
  • Cayuga
  • SASE+
  • CommonSens
  • …

• Open source:
  • Esper

• Commercial:
  • Aleri
  • StreamBase
  • System S
  • And more (see complexevents.com)!
Online Analysis of Myocardial Ischemia From Medical Sensor Data Streams with Esper

Stig Støa¹, Morten Lindeberg², Vera Goebel²

¹ The Interventional Centre (IVS), Rikshospitalet University Hospital, Oslo, Norway
² Distributed Multimedia Systems, Department of Informatics, University of Oslo, Norway
Adaptive Sized Windows To Improve Real-Time Health Monitoring – A Case Study on Heart Attack Prediction

- **Application**: Real-time health monitoring
- **Problem**: DSMS mainly support windows of static size
- **Goal**: Adapt data stream processing to physiological processes (variable durations), such as heartbeats or cardiac cycle
  -> DSMS with *time-based sliding windows of adaptive size*
Idea

- External events (tuple results from external query) determine the window size of a sliding window
- ECG stream to detect heartbeats (QRS detection)
- Accelerometer stream to detect heart displacement (Ischemia detection)
- Output of QRS detection (delay) determines when to trigger the flushing of the sliding window in Ischemia detection query
- ‘Delay’ is used to slow down accelerometer stream to account for QRS detection delay in the FIFO queue
Experiment Goal #1

• Recreate off-line technique (Elle et al. 2005) conducted in MATLAB
• Early recognition of regional cardiac ischemia
• 3-way accelerometer placed on left ventricle of the heart

• **Single metric:**
  • Fast Fourier Transformation (FFT) is used to examine the accelerometer signal in the frequency-domain
  • Euclidian distance vector \( (EDV(i)) \) between reference vector \( RV(0) \) and current vector \( CV(j) \), where \( j \) is the latest sample number
  • \( CV(j) \): FFT over sliding window (size 512 over y-axis)
  • \( RV(0) \): FFT over baseline window (first 512 samples)

• Data set from surgery performed on pigs at the Interventional Centre
• We can conduct experiments with the same data set (data set 1)
Experiment Goal #2

- Improve results by adding beat-to-beat detection using a QRS detection algorithm on ECG signals
  - Each ECG trace of a normal heartbeat typically contains a QRS event
  - A good reference for separating heartbeats
- We need to perform FFT over sliding windows of variable size!
- Cannot use the same data, use new data set that include ECG (data set 2)
Challenges

1. Incorporate signal processing operations
   • Problem: Not supported in the query language
   • Fast Fourier Transformation of the accelerometer signals
   • Euclidian distance vector from baseline window
   • QRS detection for detecting the heartbeats from the ECG signals
   • Solution: Custom aggregate functions

2. Static sized windows are not feasible for beat-to-beat detection
   • Problem: Heartbeat duration is not a static pre-known size. DSMS window techniques only describe static time-based or tuple-based windows.
   • Solution: Introduce variable length triggered tumbling windows

3. Synchronize the two streams
   • Problem: QRS detection introduces variable delay (approx. 91 samples)
   • Solution: Introduce variable buffer, that “slows” down the accelerometer stream
Signal processing operations

• Implement as custom aggregate functions
• Use defined Java interface and simply add to query engine
• Implemented methods:
  • **QRSD(v)**: QRS detection based on algorithm from Hamilton et al. 1986, source code is public available
  • **edv(v)**: Euclidian distance from baseline
Variable length triggered tumbling windows

• The ECG stream is aggregated into a stream consisting of QRS events $S_b$.
• This stream ($S_b$) triggers the flushing of the sliding window $w(t)$ where the custom aggregation over the stream $S_a$ is performed.
• This window technique is not supported by Esper => We implemented a “workaround” exploiting functionality of externally timed windows.
Stream Synchronization

- The QRS detection algorithm over the ECG stream introduces a variable delay $\Delta t$.
- Introduce the same delay to the accelerometer stream.
- Accelerometer stream is sent through a FIFO queue with dynamic size.
- QRS detection function sets the dynamic size of the FIFO queue (also triggers the flushing of the aggregate window, in order to obtain dynamic windows).
Results #1 (data set 1)

SELECT edv(y) FROM Accelerometer WINDOW LENGTH(512)

Easier than MATLAB

Occlusion occurs after 80 seconds

Perfusion after 170 seconds

Figure shows a perfect overlap, the technique by Elle et al. 2005 can be recreated online using Esper
Results #2 (data set 2)

Plot shows fixed sliding window (512 samples) and dynamic triggered window (based on QRS detection) => less variance!

SELECT edv(y) 
FROM Accelerometer 
TRIGGER WINDOW BY QRSD(ECG.value)

Occlusion

Perfusion

Sudden drop caused by ultra sound probe
Results #3 (data set 2)

The bottom plot represents local minimum value for the accelerometer stream.

Query with added local minimum value => easy to change!

SELECT edv(y), min(y) FROM Accelerometer TRIGGER WINDOW BY QRSD(ECG.value)
Implementation

- Java and Esper (open source component for event processing available at http://esper.codehaus.org/)

- Use existing window model, Esper is not changed

- Base window boundaries on the manipulated timestamps (registered as external timestamps in the Esper query) calculated from external / trigger query
Conclusion

• DSMSs can be used for real-time analysis
  => easy to model and investigate new methods

• Method of online analysis of medical sensor data focusing on detection of myocardial ischemia

• Added beat-to-beat detection by using ECG
  -> Results with less variance

• New window type for DSMS
  -> Variable length triggered tumbling windows