Applying Drools Fusion Complex Event Processing (CEP) for Real-Time Intelligence

Adam Mollenkopf
Strategic Technologist
FedEx Custom Critical

Edson Tirelli
Sr. Software Engineer
JBoss Enterprise Middleware

Sept. 2, 2009
Applying Drools Fusion

Agenda

• Intro to Complex Event Processing (CEP)

• Stateful Rules Engine + CEP = Real-Time Intelligence
  • Drools Expert + Drools Fusion

• CEP Applied – FedEx Custom Critical Case Studies
  • Demonstration, Architecture Review, and Code Walk-Through
Complex Event Processing

*Defined*

Complex Event Processing, or CEP, is primarily an event processing concept that deals with the task of processing multiple events with the goal of identifying the meaningful events within the event cloud.

CEP employs techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes.

--- wikipedia
Event Processing

Typical Scenarios

Usually receive large volume of events, but only a small percentage are of real interest.

Individual Events are usually not important. The system is typically more concerned about patterns of related events, their relationships, and what can be inferred from them.
Characteristics of a CEP Engine

Event Processing

Support processing high volume Streams of Events.

Typically fed from **SOA endpoints** such as JMS Queues/Topics, Web Service calls, Databases, flat files, or sockets.

An **Event** is a record of state change. It is something that already happened, and the past cannot be changed, events are immutables.
Drools Fusion – CEP Applied
Stateful Rules Engine + CEP = Real-Time Intelligence

FedEx Custom Critical Case Studies

Dispatch Eligibility  Capacity Allocation Management
Accurate-ETA  En-Route Tracking
FedEx Custom Critical En-Route Tracking

CEP Applied
FedEx Custom Critical En-Route Tracking
CEP Applied
FedEx Custom Critical En-Route Tracking

CEP Applied

Vehicle Event Stream
Aircraft Event Stream
Shipment Event Stream

New Facts

Inferred Facts

Reasoning

Knowledge Based Reasoning
CEP [Temporal Reasoning]
Spatial Reasoning

Knowledge Bases
FedEx Custom Critical En-Route Tracking

CEP Applied
FedEx Custom Critical En-Route Tracking

CEP Applied


Shipment Event Stream

New Facts

Inferred Facts

Reasoning

Knowledge Based Reasoning

CEP [Temporal Reasoning]

Spatial Reasoning

Knowledge Bases
Conditional Expressions

‘from’ can work on any expression, not just a nested field on a bound variable.

Using ‘from’ with Hibernate Named Queries

```java
rule "Find Vehicles for a given zip code"
when
  $zipCode : ZipCode()
  Vehicle() from $hibernate.getNamedQuery( "FindVehicles" )
    .setParameters( [ "zipCode" : $zipCode ])
    .list()
then
  Hibernate session
end
```
Conditional Expressions

accumulate

Accumulating Values

rule "accumulate"
when
  $acc : Number( intValue > 100 ) from accumulate
      ( Vehicle( kind == "E", $s : shipmentCount )
        sum( $s ) )
then
  print "Tractor/Trailer shipment count is " + $acc.sum;
end
### Patterns and CE Chaining with ‘from’

```java
rule "collect"
when
    $zipCode : ZipCode()
    $acc : Number( intValue > 100 ) from accumulate
(( Bus( kind == "E", $s : shipmentCount )
  from $hibernate.getNamedQuery( "FindVehicles" )
  .setParameter( [ "zipCode" : $zipCode ])
  .list(), sum( $s ) )
then
    print "Tractor/Trailer shipment count for " + $zipCode + " is " + $acc.sum;
end
```
Concurrent Event Stream Processing
Misperception 1: Rules Engines do not scale for CEP

Rules Engines have historically had a single point of insertion. CEP allows for concurrent Streams of Events.

Patterns evaluate facts sequentially in a single thread.
Using entry-point in a Rules-Engine for CEP

EntryPoint ep = session.getEntryPoint("ShipmentEventsEP");
ep.insert(event);  // Now we can insert different streams concurrently

rule “Process Scheduled Pickup from Entry Point"
  when
    $c : Customer(type == "VIP")
    ScheduledPickupEvent(customer == $c) from entry-point "ShipmentEP"
  then
    ...
  end

- Patterns can now optionally specify their entry-point
- When not specified the “default” entry-point is used
Declaring a type as an Event

declare VehiclePositionEvent  
  @role( event ) 
end; // will be retracted when it is no longer needed

declare VehiclePositionEvent  
  @role( event )  
  @timestamp( timestampAttr )  
  vehicleId : String  
  longitude : double  
  latitude : double  
  timestampAttr : long  
end;
Temporal Operators

Reasoning over Time

Misperception 2: **Rule Engines do not have a rich enough set of temporal comparison operators.**

---

**Temporal Operator ‘after’ Detection**

```check
rule "Confirm Pickup occurs within Pickup standard"
when
  $c : Customer( type == "VIP" )
  $spe : ScheduledPickupEvent( customer == $c )
  from entry-point "ShipmentEP"
  PickupEvent( relatedEvent == $spe.id, this after[0m, 30m] $spe )
  from entry-point "VehicleEP"
then
  ...
end
```

PickupEvent must occur between 0 and 30 minutes ‘after’ ScheduledPickupEvent
Temporal Operator Not ‘after’ Detection

rule "Detect Pickups that have not occurred within Pickup standard"
when
$c : Customer( type == "VIP" )
$spe : ScheduledPickupEvent( customer == $c )

from entry-point "ShipmentEP"
not PickupEvent( relatedEvent == $spe.id, this after[1s, 10s] $spe )

from entry-point "VehicleEP"
then
... Existing Drools ‘not’ Conditional Elements can be used to detect non-occurrence of events
end
Temporal Operators
Reasoning over Time

Allow for detection, correlation, aggregation, and composition of events. Temporal Constraint operators express relationship between events.

<table>
<thead>
<tr>
<th>Temporal Constraint Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>coincides</td>
</tr>
<tr>
<td>before</td>
</tr>
<tr>
<td>after</td>
</tr>
<tr>
<td>meets</td>
</tr>
<tr>
<td>metby</td>
</tr>
<tr>
<td>overlaps</td>
</tr>
<tr>
<td>overlappedby</td>
</tr>
<tr>
<td>during</td>
</tr>
<tr>
<td>includes</td>
</tr>
<tr>
<td>starts</td>
</tr>
<tr>
<td>startedby</td>
</tr>
<tr>
<td>finishes</td>
</tr>
<tr>
<td>finishedby</td>
</tr>
</tbody>
</table>
# Temporal Operators

## Reasoning over Time

<table>
<thead>
<tr>
<th></th>
<th>Point-Point</th>
<th>Point-Interval</th>
<th>Interval-Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>A before B</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>A meets B</td>
<td><img src="image4" alt="Diagram" /></td>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td>A overlaps B</td>
<td><img src="image7" alt="Diagram" /></td>
<td><img src="image8" alt="Diagram" /></td>
<td><img src="image9" alt="Diagram" /></td>
</tr>
<tr>
<td>A finishes B</td>
<td><img src="image10" alt="Diagram" /></td>
<td><img src="image11" alt="Diagram" /></td>
<td><img src="image12" alt="Diagram" /></td>
</tr>
<tr>
<td>A includes B</td>
<td><img src="image13" alt="Diagram" /></td>
<td><img src="image14" alt="Diagram" /></td>
<td><img src="image15" alt="Diagram" /></td>
</tr>
<tr>
<td>A starts B</td>
<td><img src="image16" alt="Diagram" /></td>
<td><img src="image17" alt="Diagram" /></td>
<td><img src="image18" alt="Diagram" /></td>
</tr>
<tr>
<td>A coincides B</td>
<td><img src="image19" alt="Diagram" /></td>
<td><img src="image20" alt="Diagram" /></td>
<td><img src="image21" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Temporal Reasoning
Sliding Time Windows

Misperception 3: **Rule Engines react to events happening now.**

\[
\text{TemperatureRead( vehicleId = "E1000" ) over window:time ( 10m )}
\]
\[
\text{TemperatureRead( vehicleId = "E1000" ) over window:length ( 10 )}
\]
Temporal Reasoning
Aggregations

Misperception 4: Rules Engines do not deal with aggregations

rule “Average temperature reading for vehicle E1000 over last 10 minutes”
when
  $n : \text{Number}( \text{intValue} > 8 )$
  from accumulate ( $tr : \text{TemperatureRead}( \text{vehicleId} = \text{“E1000”} )$
  over window:time ( 10m ),
  average( $tr.temperatureValue$ ) )
then
  ...
end
Applying CEP at FedEx Custom Critical

Declaring point-in-time Vehicle Event

```java
import com.fedex.enroutetracking.VehicleSensors;
declare VehicleSensors
    @role( event )
    @timestamp ( lastEventTime.time )
    @expires ( 1h30m ) // of the form: [#d] [#h] [#m] [#s] [#ms]
end
```

Declaring interval-based TrafficIncident Event

```java
import com.fedex.tracking.traffic.TrafficIncident;
declare TrafficIncident
    @role( event )
    @timestamp ( startTime )
    @duration ( incidentDuration )  // in milliseconds
end
```
Applying CEP at FedEx Custom Critical

Producing Events to an Entry-Point

KnowledgeBase kb = readKnowledgeBase();
StatefulKnowledgeSession ks = kb.newStatefulKnowledgeSession();
WorkingMemoryEntryPoint ep = ks.getWorkingMemoryEntryPoint("VehicleEP");
Vehicle vehicle = …;
vehicleEP.insert(vehicle);

Consuming Events from an Entry-Point

rule "Revise Stop ETAs based on new Vehicle Position"
when
    $vehicle : Vehicle ( $vid : vehicleId, $pos : position,
    status == "EnRouteToStop" ) from entry-point "VehicleEP"
    $stops : Stops ( vehicleId == $vid )
then
    $stops = EtaCalculator.reviseStopETAs( $vehicle, $stops );
    update( $stops );
end
Detecting Idle (Slow Moving) Vehicles

**rule** “Speed Idle En-Route Alert”

**when**

$stats : SensorStatistics( $vid : vehicleId )
$alerts : Alerts( vehicle == $vid )
$vehicle : Vehicle ( vehicleId == $vid, inStopProximity == false )
Number ( $avgSpeedMph : floatValue < 15 )

from accumulate (VehicleSensors( vehicleId == $vid, inStopProximity == false, $speedMph : speedMph )

over window:time ( 15m ) from entry-point “VehicleEP”,
average ( $speedMph ) )

**then**

$alerts.addAlert( new Alert (… ) );
update ( $alerts );

end

Accumulator functions: sum, count, min, max collect, [custom defined]
Applying CEP at FedEx Custom Critical
Deterministic Simulation Testing (Pseudo-Clock)

Reasoning over time requires a Reference Clock. Session Clock implements the GoF Strategy Pattern. Rules Testing requires a controlled environment of input rules. To Replay or Simulate scenarios it is necessary to control the Flow of Time via the Pseudo Clock. Regular Execution requires a Real-Time Clock.
En-Route Tracking Situational Awareness

Demonstration,
Architecture Review,
and Code Walk-Through
Applying CEP at FedEx Custom Critical
Capacity Allocation Management Case Study
Applying CEP at FedEx Custom Critical
Capacity Allocation Management Case Study
En-Route Tracking
Architecture Review
& Code Walk-Through
En-Route Tracking
Architecture Review & Code Walk-Through
En-Route Tracking
Architecture Review
& Code Walk-Through

EnRouteEventReceiver
+ onMessage (msg: Message) : void
+ synchronized get...Events() : ArrayList<...>

FlexPublisher
- msgBroker : MessageBroker
+ publishFlexEvent (body: Object) : void

TrackingState
- trackingCache : Map<String, Tracking>
+ updateTracking (t: Tracking) : void
+ getTracking (vehicleId: String) : Tracking

TrackingListener
+ run() : void
- getLatestFromReasoning() : Trackings

EnRouteTracking
- eventReceiver : IEventReceiver
- knowledgeBase : kBase
- kSession : StatefulKnowledgeSession
  enroutetracking.drl
+ run() : void
+ handleEvents() : void
+ get...() : HashMap<String, ...>

Alert / Activity Reasoning
Event Processing

Spring

RIAs
Flex Push (RTMP/AMF)
Flex Remoting (AMF)

Data Management
Observables

ITrackingState

SpringInitializingBean
Singleton
Thread

JMS Topic

Drools
Expert Fusion

Servlet

Event Streams
Simulator / Simulations
Playbacks
Live Feeds

JMS Topic

SpringInitializingBean
IEventReceiver
MessageListener

EnRouteEventReceiver

entry-points
Stateful Rules Engine + CEP = Real-Time Intelligence
Where to Learn More

Complex Event Processing

“The Power of Events” by David Luckham
See also http://complexevents.com

Vendor Documentation
Drools Fusion, Drools Expert
http://www.jboss.org.drools.documentation.html
http://www.jboss.com/drools
http://blog.athico.com

Our Contact Information:
Adam.Mollenkopf@fedex.com
ETirelli@redhat.com
QUESTIONS?

TELL US WHAT YOU THINK:
REDHAT.COM/JBOSSWORLD-SURVEY
FOLLOW US:
TWITTER.COM/REDDHATSUMMIT

TWEET ABOUT US:
ADD #SUMMIT AND/OR #JBOSWWORLD TO THE END OF YOUR EVENT-RELATED TWEET