In this chapter, we discuss the design and implementation of integrating complex event processing with a stream processing system. In Chapter 8, we derived the need for this integration and proposed a three-phase processing model (shown in Figure 8.4). For this synergistic integration, the stages, relevance, as well as the significance of each stage has been discussed in Section 9.4 and illustrated in Figure 9.2. In addition, meaningful extensions, of stream and complex event processing for enhancing the functionality of the overall system, have been elaborated in Sections 8.4.1, 9.5, and 9.6.

In Figure 9.2, stage 1 corresponds to a general purpose DSMS (whose implementation has been discussed in Chapter 10), and stage 3 corresponds to a CEP. Usually, rule processing (stage 4) is part of the CEP system. Stage 2 is the interface that bridges the two and creates a system that can be used in multiple ways – as a stream processing system, a CEP system, and a combination thereof – with capabilities to associate CQs with events and vice versa in a flexible manner. Complex applications with multiple stages of stream and event processing can be readily accommodated using the architecture presented in Figure 9.2. Hence, in this chapter we focus on stage 2 and extensions of CEP.

Rather than integrating complex event processing into MavStream from scratch, we adapt an existing, complex event processing subsystem to obtain a versatile system that can be used in multiple ways as indicated above. Although this chapter discusses the integration of two specific, independently developed systems – Snoop-based Local Event Detector (or LED\(^1\) described in [88, 187, 253, 258, 351]) and the MavStream stream processing system described in Chapter 10, we believe that the integration approach presented here is applicable to arbitrary stream and CEP systems provided both use the same computation model (dataflow in this case).

---

\(^1\) In the rest of the book, the LED, which uses the Snoop event specification language, is used interchangeably with a generic complex event processing system or CEP.
To recapitulate, computed events from continuous queries act as primitive events. Event operators supported by the LED include: and (occurrence of two events in any order), sequence (occurrence of two events in a particular order), not (non-occurrence of an event within a well-defined time interval or within a time interval defined by two arbitrary events), or (occurrence of one of the two events), temporal (a clock-based event which occurs at a specified, future time), aperiodic (non-periodic occurrence of an event within a well-defined time interval or within a time interval defined by two arbitrary events), periodic (repeated occurrence of a temporal event with constant and regular time interval within a well-defined time interval or within a time interval defined by two arbitrary events), frequency or cardinality (minimum number of times an event should occur), and plus (a temporal event based on another event with a time offset). Complex events can be composed using computed events (used as primitive events) and any of the above operators recursively. LED also supports event consumption modes (recent, chronicle, continuous, cumulative, and proximal-unique) briefly discussed in Section 9.2.2. LED also supports a prioritized rule execution model in immediate or deferred coupling modes briefly introduced in Section 9.2.3. Cascaded event and rule processing are also supported in the LED.

This chapter starts with the integration issues in Section 11.1. We then present the design and implementation of the integrated system in Sections 11.2 and 11.3. Implementation of stream modifiers introduced in Sections 8.4.1 and 9.5.2 is continued in Section 11.4. Finally, we present some of the additional benefits (e.g., starting and stopping continuous queries based on time or event occurrences) by using the available CEP component in Section 11.5. We conclude in Section 11.6 and outline features that are being added currently.

11.1 MavEStream: Integration Issues

We use the car accident detection and notification (Example 9.1 discussed in Section 9.1) to highlight the issues that arise during the integration of stream and complex event processing systems. As explained in the example, an accident is detected when Immobility, Speed Reduction are computed and their Correlation is detected.

The CQ that detects whether the car has stopped (i.e., Immobility) from Section 9.1 is repeated below.

```sql
CREATE CQ CQ_Immobile AS 
  SELECT carId, Xpos, count(*) 
  FROM CarLocStr [Range 2 minutes] 
  GROUP BY carId, Xpos 
  HAVING count(*) > 3
```

CREATE CQ CQ_Immobile AS ( 
  SELECT carId, Xpos, count(*) 
  FROM CarLocStr [Range 2 minutes] 
  GROUP BY carId, Xpos 
  HAVING count(*) > 3
)