ACTIVE MONITORING USING REAL-TIME METRIC LINEAR TEMPORAL LOGIC SPECIFICATIONS

Gabor Simko¹ and Janos Sztipanovits¹

¹Institute for Software Integrated Systems, Vanderbilt University, 1025 16th Ave S, Nashville, USA
{gabor.simko, janos.sztipanovits}@vanderbilt.edu

Keywords: Real-time Monitoring: Temporal Logic: Active Monitoring: Policy Monitoring: Control Design.

Abstract: Monitoring temporal relationships among events in event streams has wide scale applicability in health information systems. From detecting violations of privacy policies in message sequences to diagnosing conditions in physiological data streams real-time event monitoring of temporal invariants is becoming an important tool for system design. We developed an Active Real-Time Event Monitoring and Integration System (ARTEMIS) capable of integrating event streams and monitoring the existence of temporal invariants among events expressed in a safety fragment of metric first-order temporal logic (MFOTL). The paper discusses the mathematical foundations of the monitor, and demonstrates the application concepts in a physiological alarm generator and clinical information workflow system.

1 INTRODUCTION

The concept of policies is a widely used abstraction in health information system design. Policies may be defined with different purposes. For example, privacy policies (Bartha, Datta, and Mitchell, 2006) express restrictions on information flows among actors of a care delivery environment. Alert policies define the rules for signalling alerts in clinical environments. Treatment policies capture the decision rules for applying and ordering treatment activities. These policies share some common characteristics, namely, all of them can be modelled by formal logic and most of the time they contain temporal relationships. Huge difference is found however in time scales (from seconds to years), and whether they are only passive monitors or active integrators of event streams. Policies defined for passive monitoring express logical and temporal invariants over event streams. The passive monitors observe the event sequences and indicate if the invariants are violated. However, most treatment policies and several alert policies need active participation, for generating new events and integrating policy groups via the generated events (Figure 1). This is necessary for requesting actions (e.g. approvals in privacy policies or tests in treatment policies) that may be time and resource consuming and must be scheduled only by demand.

The request events \{r₁, ..., rₘ\} and the outcome of the requested activities \{t₁, ..., tₘ\} may be used for integrating other policies in the monitoring process. The interplay between the different policies leads to the concept of Active Real-Time Event Monitoring and Integration System, (ARTEMIS) which monitors different events, displays or logs policy violations and starts up activities, which may also affect other policies.

Figure 1: Schema of Active Real-Time Monitoring System: large arrows denote the access points to external resources, small arrows represent data flow. Monitor can activate Tester Unit on demand, and the feedback mechanism is provided to the Monitor.

Policies may be modelled with two different approaches, rule-based and statistical based. Due to space limitations we do not discuss statistical based
methods here. Temporal logic is often used to describe rule-based policies (Basin, Klaedtke, and Muller, 2010) and to monitor events in a rule-based fashion. Our choice based on algorithmic considerations fall on a subset of temporal logic called Metric First Order Temporal Logic (MFOTL) (Basin, Klaedtke, Muller and Pfitzmann, 2008). MFOTL can be used to specify a broad set of complex temporal constraints, while real-time operation and reasonable computation complexity is achievable. We built our monitor based on the concepts defined in their work, but over an extended MFOTL language and using a significantly different monitoring algorithm.

The structure of the paper is the following: in Section 2 we introduce MFOTL language and MFOTL monitoring. Section 3 details the concept of ARTEMIS, and in section 4 we show a clinical workflow example. Finally, in section 5 we give the conclusions.

2 BACKGROUND

2.1 Syntax of MFOTL language

We define the logic MFOTL as a superset of MFOTL (Basin et al., 2008), such that point and interval based semantics are both supported. MFOTL may also be considered as the union of metric temporal logic (MTLΩ) (Alur and Henzinger, 1991, Kojymans, 1990) and metric interval temporal logic (MITL[α,β]) (Nickovic and Maler, 2007, Alur, Feder, and Henzinger, 1991) extended with predicates and quantification. The past temporal modalities are interpreted on possibly infinite intervals, while the future temporal modalities are bounded. Description of the MFOTL language is based on the work of Basin et al. (2008) and Nickovic and Maler (2007).

Formulae of MFOTL are inductively defined in Backus-Naur Form by the following grammar:

\[ p := t | \neg p | p_1 \land p_2 | \exists x. p | p_1 U p_2 | p_1 S p_2, \]

where \( I \) is a possibly singular time interval and \( t \) is a basic term, i.e. a function compared to a constant value, or a boolean predicate value. The operators represent the standard negation, conjunction, existence, until and since operators, respectively.

Based on the basic formulae we can express other standard logic operators and constants, such as true (T), false (F), disjunction (\( p_1 \lor p_2 \)) and universality (\( \forall x. p \)). Also, we can express release and trigger operators, eventually and always operators, and their past versions, once and historically:

\[ p_1 R p_2 := \neg (\neg p_1 U \neg p_2) \]
\[ p_1 T p_2 := \neg (\neg p_1 S \neg p_2) \]
\[ \Diamond p := U \top p \]
\[ \Box p := F I p \]
\[ \diam p := \neg S I p \]
\[ \square p := F T I p \]

Here we illustrate the MFOTL language with a frequently arising example: the policy declares that a patient \( p \) had to give their consent in the last 8 days to disclose their lab results prior to the disclosure taking place. The purpose of the monitor is to detect if illegal disclosure happened. The policy is easily expressed using MFOTL:

\[ \text{isclosed}(x) \rightarrow \mbox{[8days]} \text{consent}(x) \]

If this expression is not satisfied at any time \( t \) with respect to patient \( x \), the policy was violated for that patient.

2.2 MFOTL Monitoring Algorithm

The main idea behind online monitoring is to incrementally build an inner representation of previous states without storing all the unnecessary details. A possible solution is to introduce auxiliary relations describing these past states. Satisfiability in the current state is answered by evaluating only these additional relations, i.e. answering the satisfiability of a first-order logic expression. Our monitoring algorithm works by transparently building and evaluating these auxiliary relations.

3 ARTEMIS ARCHITECTURE

ARTEMIS systems are built from three different kinds of components (Figure 1), sources, testers, and monitor. Sources are independent event generators (e.g. measurement devices or audit systems), which set up relations and functions used by the MFOTL monitor. The sources send all the necessary information to the monitor, which automatically extracts and stores the relevant data. The monitor continuously checks whether the policies are satisfied and on satisfaction the appropriate testers (e.g. treatment procedures, lab tests) are activated. Finally, the results of testers are fed back to the monitor, which may lead to other coupled actions.

The controller actions are declared in the form of Horn clauses (\( \text{head}(x) \leftarrow \text{body}(x) \)), where the head is the action and body is the conditions leading to the action. We can define any number of such expressions as long as the body of the Horn clause is temporal sub-formula domain independent (Basin et al., 2008). The expressions are ordered which
defines the order of evaluation. Expressions may refer to each other: any expression may use the past values of any other expression including themselves or the current results of any expressions defined earlier in the order.

ARTEMIS shows several advantages over systems built with low level languages (Table 1). The fact that policies are well represented in formal logic leads to a compact, readable and easily maintainable policy code in ARTEMIS. The optimized monitoring algorithm results in high performance and optimal resource management. Furthermore, extensibility is easily achieved by using formal logic conjunction and disjunction operators without touching any of the previously written code.

Table 1: Advantages of ARTEMIS.

<table>
<thead>
<tr>
<th></th>
<th>ARTEMIS with MFOTL monitoring</th>
<th>Traditional system built with a low level language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy code complexity</td>
<td>Very compact</td>
<td>Highly complex</td>
</tr>
<tr>
<td>Performance</td>
<td>Automatically high, optimal</td>
<td>High, if optimized</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Easily extensible</td>
<td>Extensible, but complicated</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Easy maintenance</td>
<td>Cumbersome maintenance</td>
</tr>
</tbody>
</table>

Comparison to rule-based workflow management systems like Drools does not show this great difference. The main advantage of ARTEMIS is its MFOTL engine which can evaluate temporal logic with complex temporal expression. Although Drools is very efficient to describe simple temporal policies, it cannot handle compound temporal operators, which may seriously limit its applicability in some scenarios (e.g. signal processing).

4 EVALUATION

We evaluated ARTEMIS using the initial phases of the sepsis alert and treatment protocol (Figure 2). By monitoring several physiological data of patients we could express the Systemic Inflammatory Response Syndrome (SIRS) alert system (Shapiro, Howell, Talmor et al., 2006). After SIRS alert was issued, ARTEMIS had to request the approval of a doctor to begin the sepsis treatment protocol. In case the approval arrived, a lab test request was issued to analyse additional conditions. Only after the receipt of approval and lab tests could the sepsis treatment start (Dellinger, Levy, Carlet et al., 2008).

The SIRS alert protocol (Shapiro et al., 2006) defines validity ranges for physiological data. In case the measured function is outside the validity range, the measured data is abnormal. Abnormal body temperature and white-blood cell count are major criteria, abnormal respiration rate and heart rate are minor criteria. The protocol defines two kinds of alerts: high priority alerts are issued if two major criteria were met in the last 24 hours; low priority alerts are issued if at least one major and one minor criterion were met in the last 24 hours, and no alert were issued in the last 24 hours.

Our system contained four measured functions interpreted on patients: temp (temperature), wbc (white-blood cell count), rr (respiratory rate) and hr (heart rate). Using these functions we could express the policy as seen in Table 2. The derived abnormal functions were satisfied for a patient x, if their measurement was abnormal (i.e. out of normal range). majorCriteria was true, if at least one major criterion held, minorCriteria was true if at least on minor criterion held. Based on these criteria we could define highPriority and lowPriority, which were satisfied when the high priority or low priority alert requirements (see above) were met. issueHighAlert and issueLowAlert signalled, when a
high priority or low priority alert had to be sent out for patient x. For any patient x, when we reached SIRS alert, we initialized the Verification tester (once in every 24 hours, because the test takes significant amount of time; also note that this is not part of the original protocol, we used it for demonstration purposes only) for that patient. Relation approval signalled the results of the verification. If approval was received, the patient was sent to lab tests by issuing Sepsis lab. On the arrival of lab tests, relation labtests done was updated. If lab tests were done within 24 hours after the approval of sepsis treatment, we could start the sepsis treatment action signalled by Sepsis treatment.

Even though the performance of the algorithm heavily depends on several factors, we simulated SIRS alert with 100000 distinct events affecting 100 patients leading to 25074 issued SIRS alert to demonstrate the order of magnitude. The average performance was 0.21ms / event.

### 5 CONCLUSION

We showed the concept of Active Real-Time Event Monitoring and Integration System (ARTEMIS) which is an extension of traditional real-time monitoring systems with active participation from the part of the monitor. As a workflow management system ARTEMIS competes with systems like Drools, but supports a broader and more expressive set of temporal expressions which might show immediate advantages in signal processing scenarios.

### REFERENCES


<table>
<thead>
<tr>
<th>abnormalWbc(x)</th>
<th>wbc(x) &lt; 4000</th>
<th>wbc(x) &gt; 20000</th>
</tr>
</thead>
<tbody>
<tr>
<td>abnormalTemp(x)</td>
<td>temp(x) &lt; 96.8</td>
<td>temp(x) &gt; 100.4</td>
</tr>
<tr>
<td>abnormalRR(x)</td>
<td>rr(x) &gt; 20</td>
<td></td>
</tr>
<tr>
<td>abnormalHR(x)</td>
<td>hr(x) &gt; 90</td>
<td></td>
</tr>
<tr>
<td>majorCriteria(x)</td>
<td>abnormalWbc(x) ∨ abnormalTemp(x)</td>
<td></td>
</tr>
<tr>
<td>minorCriteria(x)</td>
<td>abnormalRR(x) ∨ abnormalHR(x)</td>
<td></td>
</tr>
<tr>
<td>highPriority(x)</td>
<td>![logical expression]</td>
<td></td>
</tr>
<tr>
<td>lowPriority(x)</td>
<td>![logical expression]</td>
<td></td>
</tr>
<tr>
<td>issueHighAlert(x)</td>
<td>![logical expression]</td>
<td></td>
</tr>
<tr>
<td>issueLowAlert(x)</td>
<td>![logical expression]</td>
<td></td>
</tr>
<tr>
<td>SIRSAlert(x)</td>
<td>![logical expression]</td>
<td></td>
</tr>
<tr>
<td>Sepsis_lab(x)</td>
<td>approval(x)</td>
<td></td>
</tr>
<tr>
<td>Sepsis_treatment(x)</td>
<td>![logical expression]</td>
<td></td>
</tr>
</tbody>
</table>