A Temporal Abductive Diagnostic Process for Runtime Properties Violations

Theocharis Tsigkritis[1] and George Spanoudakis[1]

[1] *Department of Computing, City University London, UK*
{t7t, G.Spanoudakis}@soi.city.ac.uk

Abstract. The monitoring of properties of complex software systems can provide the core functionality for detecting violations of such properties. However, the violations detection cannot be always sufficient for the preservation of the properties. Except for the detection, the explanations of the occurrence of a violation could play significant role for the preservation task. In particular, diagnosis can indicate the cause(s) of a violation. Thus, diagnostic information is necessary for preserving the properties due to the support that can provide for deciding on the appropriate countermeasure against a violation. In this paper, we describe a process for diagnosing runtime violations of properties that we have developed as part of a runtime monitoring framework. The process is based on a combination of abductive, temporal and evidential reasoning over violations of process properties expressed in Event Calculus.

Keywords: Abductive reasoning, temporal constraints problem, Dempster-Shafer theory of evidence, Event Calculus.

1 Introduction

Monitoring properties of software systems at runtime is widely accepted as a technique for increasing the resilience to dependability failures and security attacks and several approaches have been developed to support it (see [7] for a survey). Although basic monitoring provides mechanisms for detecting violations of such properties, it cannot always provide the information that is necessary in order to understand the reasons that underpin the violation of a property and decide what would be an appropriate reaction to it.

To appreciate the problem, consider the case of an Air Traffic Management System (ATMS), which consists of components (radars) that monitor the traffic in different air spaces. By monitoring the operations of an ATMS at runtime, the availability and integrity of its components (e.g. radars), and the information generated by and/or exchanged between them might be ensured. For instance, a property that can be monitored in an ATMS is a property requiring that in cases where there are more than one radars covering a particular airspace and one of these radars sends a signal indicating that an airplane is in the relevant airspace, every other radar that covers the same space should also send a signal indicating the presence of the plane in it and this should happen within a certain time period after the receipt of the initial signal.

In cases where this property is violated, knowing about the occurrence of the violation itself is not sufficient for establishing the reasons why some radar has sent a signal but the other has not. Clearly getting diagnostic information about these reasons would be necessary for taking appropriate action as the violation may have been due to different reasons, including the following:

- The radar that did not send the expected signal was malfunctioning.
- The communication link between the radar that did not send the expected signal and the monitor was malfunctioning or an intruder captured the signal and prevented it from reaching the monitor.
- The radar that sent the expected signal was malfunctioning or its identity was faked by an intruder which sent a fake signal to the monitor.

Thus, identifying the reason for the violation is important for taking actions that could restore the integrity of the operation of the ATMS.

In this paper, we present a diagnosis tool that we have developed as part of the monitoring framework described in [16]. This framework has been developed within the European integrated research project SERENITY to support the monitoring of security and dependability properties in distributed and dynamically evolving systems. The implemented monitoring framework supports the specification and monitoring of properties expressed in Event Calculus (EC) [15] as rules.

In particular, we present a newly developed extension of this framework supporting the diagnosis of rule violations. The provision of diagnostic information is based on is the generation of all the possible alternative *explanations* of the events which are involved in the violations of rules, and the assessment of the plausibility of these explanations based on whether their effects correspond to events recorded during the operation of the monitored system. The key characteristic of our approach is the use of abductive reasoning [2][9][10] for the generation of explanations and belief based reasoning [14] for the assessment of explanation plausibility.

The rest of this paper is structured as follows. In Section 2, we provide a brief overview of the monitoring toolkit. In Section 3, we describe the different stages of the diagnostic process. In Section 4, we overview related work and, finally, in Section 5, we present conclusions and directions for future work.

2 Monitoring framework

The core of our monitoring framework is a generic engine for checking violations of properties expressed as EC rules of the form *body* $\Rightarrow$ *head*. The meaning of a rule is that if its body evaluates to true, its head must also evaluate to true. EC is a first-order metric temporal logic language which can be used for representing and reasoning about *events* and their effects on the state of a system over time. Our monitoring framework rules are defined in terms of the standard EC predicates. These include the predicates (i) $Happens(e,t,\Re(lb,ub))$ which denotes that an instantaneous event e occurs at some time t within the time range $\Re(lb,ub)$, (ii) $HoldsAt(f,t)$ which denotes that a state (aka fluent) f holds at the start of the execution of a system and at time t, respectively, (iii) *Initiates(e,f,t)* and *Terminates(e,f,t)* which denote the initiation or termination of a fluent f by an event e at time t respectively, and (iv) *Initially(f)* which denotes that a fluent holds at the start of the operation of a system.

An example of a rule is:

Rule 1: **Happens**(signal(_r1, _a, _s),t1,R(t1,t1)) ∧ **HoldsAt**(covers(_r1,_s),t1) ∧ (∃_r2) **HoldsAt**(covers(_r2,_s), t1) ⇒ **Happens**(signal(_r2,_a,_s), t2 ,R(t1, t1+5))

This rule expresses the condition about the radars of ATMS that we discussed in the introduction and will be violated if there is only a *signal* event form one of the two radars of ATMS that covers a specific airspace but not from the other radar.

3 Diagnostic process

As shown in Figure 1, the overall process of diagnosing the causes of rule violations includes four stages, namely:

1. *explanation generation* in which all the *possible explanations* for the individual events that were reported to the monitor and have caused the violation (referred to as "violation observations" henceforth) are generated.
2. *explanation effect identification* in which the possible consequences (effects) of the explanations of the violation observations are derived by deduction
3. *plausibility assessment* in which the effects of explanations are checked against the event log of the monitor to see if there are events that match them and could provide supportive evidence for the explanations
4. *diagnosis generation* in which an overall diagnosis for the violation is generated from the individual explanations

The generation of explanations and their effects in stages (i) and (ii) above is based on a incomplete model of the behaviour of the monitored system that is expressed in the form of EC formulas called *assumptions*. In the following, we discuss the stages of the diagnostic process in detail.

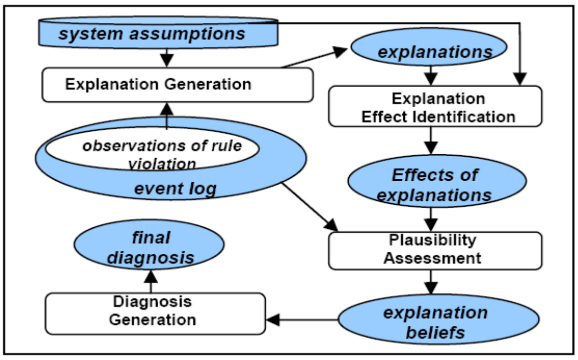

Fig. 1. Diagnostic process

3.1 Explanation generation

The generation of explanations for violation observations is based on abductive reasoning. More specifically, given a set Ω of events and fluents that are involved in the violation of a monitoring rule, this stage of the diagnostic process tries to find a set of *explanation formulas* Φ which, in conjunction the set of the *assumptions* about the system that is being monitored and the events that are known to the monitor at the time when the explanation is required (collectivelly referred to as *TH theory* in the following), entail Ω. Formally, this is a search for a set of atomic formulas Φ that satisfy the conditions:

(Cnd 1): TH $\cup \Phi$ |- Ω, and

(Cnd 2): $\forall$ *f in* Φ*: predicate (f)* $\in$ *APreds*

where *predicate (f)* is the predicate of formula *f* and *APreds* is a set of abducible predicates whose truth value can be established only by abductive reasoning.

The search for explanations is based on a newly developed algorithm (see [17]) which starts from a violation observation *P* that needs to be explained and tries to find all assumptions of the form *a: B1* $\wedge \ldots \wedge$ *Bn* $\Rightarrow$ *H* in *TH* whose head *H* can be unified with *P*. When such an assumption is found, the algorithm checks: (i) if the unification of *P* with *H* provides concrete values for all the not time variables of the predicates *B1, …, Bn* in its body, and (ii) if it is possible to derive concrete time ranges for the time ranges of all these predicates by using George Dantzig's classic *Simplex* method, which is revisited in [4]. If these conditions are satisfied, the algorithm instantiates the predicates *B1, …, Bn* and identifies which of these predicates are observable predicates (*O-preds*), deducible predicates (*D-preds*) or abducible predicates (*A-preds*), assuming that these are disjoint categories of predicates.

Then, the algorithm checks if each of the *O-Preds* and *D-preds* in the body of *a* can be matched with some recorded event or derived from the events in the monitor's log and the known system assumptions, respectively. If there are *O-preds* and *D-preds* that cannot be verified via this check, the algorithm tries to find an abduced explanation for them recursively. If such explanations are for all the non verified *O-preds* and *D-preds*, these explanations along with the *A-preds* that were determined in the current step of the explanation process are reported as the possible explanation of the initial violation observation *P*. In cases, however, where there are *O-Preds* or *D-preds* in the body of *a* that can neither be verified nor explained by abduction, the explanation generation path using *a* will fail.

As an example of explanation generation, consider again *Rule 1*. This rule would be violated by the event (E7) in the event log of Figure 2 (*Happens(signal(R1,A1,S1),7,R(7,7))* and the predicates $\neg$*Happens(signal(R2,A1,S1),t,R(7,12)),* *HoldsAt(covers(R1,S1),7)* and *HoldsAt(covers(R2,S1),7)* which can be derived from this log. More specifically, the predicate $\neg$*Happens(signal(R2,A1,S1),t,R(7,12)),* which denotes the absence of a signal from radar *R2* in the time range from T=7 to T=12, is deduced by the principle of *negation as failure* (NF) from the events (E4) and (E8) that were received from radar *R2* at T=1 and T=13 as soon as the monitor receives (E8). This is because no other event has been received from *R2* between these two time points. Also the predicates *HoldsAt(covers(R1,S1), 7)* and *HoldsAt(covers(R2,S1), 7)* can be deduced from events (E1) and (E2) in Figure 2, which denote that radars R1 and R2 cover the airspace S1 initially, and the absence of any event signifying the repositioning of any of the two radars until the time point T=7 when the monitor receives the signal for the presence of aircraft A1 in S1 from R1 (this deduction is based on the axioms of EC [12]). To explain the violation, the predicates *Happens(signal(R1,A1,S1),7,R(7,7))* and ¬*Happens(signal(R2,A1,S1),t,R(7,12))* need to be explained individually.

(E1) **Initially**(covers(R1,S1),0) [captor-0]
(E2) **Initially**(covers(R2,S1),0) [captor-0]
(E3) **Happens**(changeOfLandingApproach(AR-a,S2),0,R(0,0)) [captor-AR-a]
(E4) **Happens**(signal(R2,A2,S2),1, R(1,1)), [captor-R2]
(E5) **Happens**(changeOfLandingApproach(AR-a,S1),2,R(2,2)) [captor-AR-a]
(E6) **Happens**(permissionRequest(A1,S1),3,R(3,3)) [captor-0]
(E7) **Happens**(signal(R1,A1,S1),7,R(7,7)) [captor-R1]
(E8) **Happens**(signal(R2,A5,S1),13,R(13,13)) [captor-R2]

Fig. 2. ATMS event log

Assuming that the following assumptions are known about the ATMS:

(A0) **Initiates**(_e1,_f),t1,R(t1,t1)) ∧ ¬∃_e2,t2: **Terminates**(_e2,_f),t2,R(t1,t2)) ⇒ **HoldsAt**(_f,t2)
(A1) **Happens**(inspace(_a,_s),t1,R(t1,t1)) ∧ **HoldsAt**(covers(_r,_s),t1) ⇒ **Happens**(signal(_r,_a,_s),t2, R(t1,t1+5))
(A2) **Happens**(inspace(_a,_s),t1, R(t1,t1)) ⇒ **Happens**(permissionRequest(_a,_s), t2, R(t1-20,t1-1))

the search for an explanation of *Happens(signal(R1,A1,S1),7,R(7,7))* will detect that this predicate can be unified with the predicate *Happens(signal(_r,_a,_s), t2, R(t1,t1+5))* in the head of assumption (A1). The unification of these two predicates will be *{_r/R1, _a/A1, _s/S1}* and the linear constraint system generated for the time variable t1 in (A1) will include the constraints $t_1 \leq 7$ and $7 \leq t_1 + 5$. Thus, since the non time variables in the body of (A1) are covered by the unification and the constraints $t_1 \leq 7$ and $7 \leq t_1 + 5$ determine a feasible time range for t1 (i.e., [2,...,7]), the conditions of the explanation generation process are satisfied and the predicate *Happens(inspace(A1,S1),t1,R(2,7))* will be generated as a possible explanation of *Happens(signal(R1,A1,S1),7,R(7,7))*. Subsequently, assuming that *Happens(inspace(_a,_s),t1,R(t1,t1))* belongs to the set of the abducible predicates *Apreds*, there will be no need for further elaboration of it.

Note, however, that as *Happens(inspace(A1,S1),t1,R(2,7))* has been generated from assumption (A1), it can be returned as an explanation only if the other instantiated predicate of the body of (A1), namely *HoldsAt(covers(R1,S1),7)*, is *True* when t1 takes values in the range R(2,7). The latter predicate, however, can be deduced from the log of Figure 2 and assumption (A0). Thus, *Happens(inspace(A1,S1),t1,R(2,7))* becomes a possible explanation of *Happens(signal(R1,A1,S1),7,R(7,7))*.

3.2 Explanation effect identification

Following the generation of explanations, the next step in the diagnosis process is the identification of the expected effects of these explanations. These consequences are needed to assess the plausibility of explanations. The assessment of explanation plausibility is based on the hypothesis that if the expected effects of an explanation match with events which have occurred and recorded during the operation of the system that is being monitored, then there is supportive evidence for the explanation. This is because the events that match its expected effects might also have been caused by it.

The identification of the expected effects of an explanation is based on deductive reasoning. Generally, for an explanation $Exp=P_1 \wedge \ldots \wedge P_n$ formed as a conjunction of abduced atomic predicates, the diagnosis process iterates over the predicates P_i that constitute it and, for each of these predicates, finds the system assumptions $B_1 \wedge \ldots \wedge B_n \Rightarrow H$ which have a predicate B_j in their body that can be unified with P_i and the rest of the predicates in its body are also *True*. For such assumptions, if the predicate H in the head of the assumption is fully instantiated and its time range is determined, H is derived as a possible consequence of P_i.

Then, if H is an observable predicate, i.e., a predicate that can be matched with recorded events, H is added to the possible effects of *Exp*. If H, however, is not an observable predicate, the effect identification process tries to generate the consequences of H recursively and, if it finds any such consequences that correspond to observable events, it adds them to the set of the expected effects of *Exp*. In this way, the diagnosis process computes the transitive closure of the effects of *Exp*.
As an example of identifying the consequences of explanations, consider again the ATMS system and suppose that, in addition to assumptions (A1) and (A2), three more assumptions are known for this system, namely:

(A3) **Happens**(inspace(_a,_s),t1,R(t1,t1)) ⇒ **Initiates**(inspace(_a,_s), inairspace(_a,_s),t1)
(A4) **Initiates**(inspace(_a,_s), inairspace(_a,_s),t1) ∧ **HoldsAt**(landing_airspace_for(_s,_arpX),t1) ⇒ **Happens**(landingRequest(_a, _arpX), t2, R(t1-10,t1))
(A5)**Happens**(changeOfLandingApproach(_arpX,_s),t1,R(t1,t1))⇒ **Initiates**(changeOfLandingApproach(_arpX,_s), landing_airspace_for(_s,_arpX),t1)

The formula (A3) above states that when an event that signifies the entrance of an aircraft *_a* in an airspace *_s* becomes known a fluent called *inairspace(_a,_s)* should be initiated to signify the presence of *_a* in *_s* unless this fluent already holds. Formula (A4) states that when an aircraft *_a* enters an airspace *_s* that is used as the final landing route for approaching an airport *_arpX* then the aircraft *_a* must have made a landing request for the particular airport within the last 10 time units before entering *_s*.

Using (A3) and (A4), it is possible to determine the expected effects of the predicate *Happens(inspace(A1,S1),t1,R(2,7)))* that was generated as a possible explanation of *Happens(signal(R1,A1,S1),7,R(7,7))*. Specifically, assuming that the airspace *S1* is the landing airspace of an airport *AR-a* then the entrance of the aircraft *A1* into *S1* should be preceded some request from *A1* to land in *AR-a* or, equivalently, that a runtime event *Happens(landingRequest(A1,AR-a), t2, R(0,6))* should have occurred. Thus, the latter runtime event would be an expected effect of the explanation *Happens(inspace(A1,S1),t1,R(2,7))*.

Formally, from *Happens(inspace(A1,S1),t1,R(2,7)))* and (A3) the predicate *Initiates(inspace(A1,S1), inairspace(A1,S1), t1)* can be deduced for *t1* in [2,…,7]. As the latter predicate, however, is not an observable predicate, the diagnosis process will try to identify whether it has any observable consequences of its own. Whilst searching for such conseuqneces, *Initiates(inspace(A1,S1), inairspace(A1,S1), t1)* can be unified with the first predicate in the body of (A4). Furthermore, the other predicate in the body of this assumption, namely the predicate *HoldsAt(landing_airspace_for(S2,AR-a), t)* can also be deduced to be *True* for the time range [2,…,7] (i.e., for t in [2,…,7]) from the event (E5) in Figure 2 and assumptions (A5) and (A0). Thus, both predicates in the body of (A4) are *True* and, therefore, the predicate *Happens(landingRequest(A1,AR-a), t2, R(0,6))* in its head can be derived from it. Assuming that *landingRequest(_a, _arpX)* is an observable event, *Happens(landingRequest(A1,AR-a), t2, R(0,6))* will be established as an expected effect of the explanation *Happens(inspace(A1,S1),t1,R(2,7)))*.

3.3 Assessment of explanation plausibility

After deriving the expected effects $\Phi_C=\{C_1,\ldots,C_L\}$ of an explanation Φ, the diagnosis process searches the event log of the monitoring framework to find events that can match these effects. In this search, a match between an event *e* in the log, which has been produced by an event captor *Captor(e)* and has a timestamp t_e, and an effect C_k (k=1,…,L) is detected only if: (i) *e* has been produced by the same event captor as the captor that C_k is expected to be produced from, (ii) *e* can be unified with C_k, and (iii) the timestamp of *e* falls within the time range of C_k.

It should be appreciated, however, that although the presence of a matching event for an expected effect of an explanation confirms that the effect has indeed occurred, the absence of a matching event for an effect at the time of the search does not necessarily mean that such an event has not occurred and, therefore, cannot cast negative evidence in the validity of the consequence. This is because there might be cases where, although an event that satisfies the conditions (i)−(iii) above may have occurred, this event might not have arrived yet at the event log of the monitoring framework due to communication delays in the “channel” between the event captor that captured the event and the monitoring framework. To cope with this problem, the search for events that match an explanation effect C_k establishes that no such events have occurred if at the time of the search there is no event *e* satisfying the conditions (i)-(iii) above, and the last known value of the clock of *Captor*(C_k) (i.e., the timestamp of the last event in the log that has arrived at the monitor from this captor) is greater than the upper boundary of the time variable of C_k.

Furthermore, there is a possibility of having effects C_k for which, although no matching event satisfying (i)-(iii) can be found at the time of the search, the last received event from the relevant captor has a timestamp that is less than or equal to the upper time boundary of C_k. Such effects cannot be confirmed or disconfirmed and, therefore, cast positive or negative evidence for Φ. To cope with this uncertainty, we use the *Dempster Shafer (DS) theory of evidence* [14] for the assessment of the plausibility of an explanation, and define the function that gives the basic probability assignment to the validity of an explanation as:

Definition 1: The basic probability of the validity of an explanation is computed by the function:

$$m_E(Valid(\Phi)) = |\Phi^{C+}| / |\Phi^{C}|$$
$$m_E(\neg Valid(\Phi)) = |\Phi^{C-}| / |\Phi^{C}|$$
$$m_E(Valid(\Phi) \vee \neg Valid(\Phi)) = |\Phi^{C} - (\Phi^{C+} \cup \Phi^{C-})| / |\Phi^{C}|$$

where

- Φ^{C+} is the set of confirmed effects of Φ, defined as $\Phi^{C+} = \{C_k / C_k \in \Phi_C$ ***and*** $\exists e.\ (e \in Log$ ***and*** $Captor(e) = Captor(C_k)$ ***and*** $t_{kLB} \leq t_e$ ***and*** $t_e \leq t_{kUB}$ ***and*** $unifier(e,C_k) \neq \emptyset)\}$
- Φ^{C-} is the set of a set of disconfirmed effects of Φ, defined as $\Phi^{C-} = \{ C_k / C_k \in \Phi^{C}$ ***and*** $\neg\exists e.\ (e \in Log$ ***and*** $Captor(e)=Captor(C_k)$ ***and*** $t_{kLB} \leq t_e$ ***and*** $t_e \leq t_{kUB}$ ***and*** $unifier(e,C_k) \neq \emptyset)$ ***and*** $lastTime(Captor(C_k)) > t_{kUB}\}$
- t_{kLB}, t_{kUB} are the lower and upper boundaries of the time range of C_k, t_e is the timestamp of the event *e*, and *lastTime(Captor(*C_k*))* is the timestamp of the last event arrived from *Captor(*C_k*)* to the monitor.

According to this definition, the probability of the validity of an explanation Φ is measured as the proportion of the effects of Φ that have been confirmed by events in the event log at time *t*. Also the probability of an explanation Φ not being valid is measured as the proportion of the effects of Φ that have been disconfirmed by events in the event log. Note that, as in general $\Phi^{C+} \cup \Phi^{C-} \subseteq \Phi^{C}$, we will also have that $m_E(Valid(\Phi)) + m_E(\neg Valid(\Phi)) \leq 1$ and, m_E is not a classic probability function. As we prove in [14], however, m_E satisfies the axioms of *basic probability assignments* in the DS theory of evidence and, can therefore, be interpreted as a function of this type.

Using m_E, the basic probability of the explanation *Happens(inspace(A1,S1),t1,R(2,7))* of the violation observation *Happens(signal(R1,A1,S1),7,R(7,7))* of *Rule-1* can be computed as follows. As discussed in Section 3.2, an expected effect of this explanation is *Happens(landingRequest(A1,AR-a),t2,R(0,6))*. Another expected effect of the same explanation is the predicate *Happens(permissionRequest(A1,S1), t2, R(0,7))*. The latter effect can be derived from assumption (A2), according to which an aircraft which enters a particular airspace at some time point *t1*, must have requested permission to enter the airspace before its entrance and no more than 20 time units prior to it.

Assuming then that the request for diagnosing the violation of *Rule-1* is made at *T=15*, a search in the event log of Figure 2 will identify that the event *Happens(permissionRequest(A1,S1),3,R(3,3))* provides confirmatory evidence for *Happens(permissionRequest(A1,S1),t2,R(0,7))* but there is no matching event for *Happens(landingRequest(A1,AR-a),t2,R(0,6))*.

Furthermore, if *Happens(landingRequest(A1,AR-a), t2, R(0,6))* refers to events which are captured and transmitted by the event captor *captor-AR-a* then at the time of the search (*T=15*), it will not be impossible to establish whether an event matching *Happens(landingRequest(A1,AR-a),t2,R(0,6))* has occurred. This is because, as shown in Figure 2, the last event received from *captor-AR-a* until *T=15* is *Happens(changeOfLandingApproach(AR-a,S1),2,R(2,2))* and, therefore, the latest known time for this captor (*lastTime(captor-AR-a))*) is 2. Thus, the basic probabilities in the validity of the explanation $\Phi = Happens(inspace(A1,S1),t1,R(2,7))$ will be: $m_E(Valid(\Phi)) = 1/2 = 0.5$, $m_E(\neg Valid(\Phi)) = 0/2 = 0$ and $m_E(Valid(\Phi) \vee \neg Valid(\Phi)) = 1/2 = 0.5$.

3.4 Diagnosis generation

Having obtained the basic probability measures in the validity or not of individual explanations, the next step in the diagnosis process is to construct an aggregate explanation of the S&D rule violation. The construction of such aggregate explanations is based on assessing the overall belief in the *genuineness* of the events that are involved in the violation. This assessment is based on the hypothesis that an event *E*, which is involved in a violation of an S&D rule, is genuine if and only if at least one of the explanations that have been generated for it is valid. Based on this hypothesis, as we show in [17], the belief in the genuineness of *E* (Gen(E)) is measured as:

$$Bel(Gen(E)) = Bel(\vee_{i=1,\ldots,n} Valid(\Phi_i)) = \Sigma_{I\subseteq\{1,\ldots,n\} and\ I\neq\emptyset}(-1)^{|I|+1}\{\Pi_{i\in I}\, m_E(Valid(\Phi_i))\} \quad \text{(F2)}$$

$$Bel(\neg Gen(E)) = Bel(\wedge_{i=1,\ldots,n} \neg Valid((\Phi_i)) = \Pi_{i=1,\ldots,n}\, m_E(\neg Valid(\Phi_i)) \quad \text{(F3)}$$

whereby Φi (i=1,…,n) are the alternative explanations of *E*

The beliefs in the genuineness of *E* and its negation which are computed by the above formulas are used to decide whether or not a violation observation is confirmed by its available explanations. In particular, the computation of *Bel(Gen(E))* and *Bel(¬Gen(E))* generates a belief range for the genuineness of *E* which, according to the DS theory [14], is:

$$[Bel(Gen(E)),\ldots, Pls(Gen(E))]$$

whereby: $Pls(Gen(E)) = 1 - Bel(\neg Gen(E))$ (F4)

The lower bound of this range is the belief in the genuineness of *E* and the upper bound of it is the maximum possible value that the belief in the genuineness of *E* can take given the belief in the non genuineness of *E*. The upper bound for the belief in the genuineness of *E* is called in the DS theory, the “plausibility” of this proposition [14].

```
Generate_Violation_Explanation(R: Instance of Violated Rule)
For each predicate P in R Do
  If P is negated Then
      Explanations = explain(¬P)
  Else
      Explanations = explain(P)
  End If
  Consequences = GenerateConsequences(Explanations)
  [Bel(P),…,Pls(P)]=ComputeBeliefRange(Consequences)
  If 1-Pls(P) < Bel(P) Then
      If P is negated Then
         UnconfirmedPredicates = UnconfirmedPredicates ∪ {P}
      Else
         ConfirmedPredicates = ConfirmedPredicates ∪ {P}
      End if
```

```
  End if
End For
[BelA(Body(R)),…,PlsA(Body(R))]=
      =ComputeBeliefRangeofPredicateConjunction(Body(R))
If 1-PlsA(Body(R) < BelA(Body(R)  Then
  If Head(R) in UnconfirmedPredicates Then
      report the head predicate of the rule as the cause of violation
  End if
Else
  If Head(R) in ConfirmedPredicates Then
      report the body predicates of the rule as the cause of violation
  End if
End if
For all P in ConfirmedPredicates Do report P as a confirmed predicate
      and provide alternative explanations of P End for
For all P in UnconfirmedPredicares Do report P as unconfirmed predicate
      and provide alternative explanations of P End for
END Generate_Violation_Explanation
```

According to the Generate_Violation_Explanation algorithm, *E* is confirmed only if *Bel(Gen(E)) > Bel(¬Gen(E))* and the final diagnosis of the violation consists of the confirmed and unconfirmed events of it and their explanations. It should also be noted that if no explanation can be generated for a violation observation, the diagnosis process attempts to find an explanation of its negation and, if this is possible, the beliefs in the genuineness of the event are calculated by using the (F4) formula and the following one:

Bel(¬Gen(E)) = Bel(Gen(¬E))) (F5)

Due to (F2)-(F5), the beliefs in the genuineness of the predicates involved in the violation of Rule-1 are calculated from the alternative explanations of the relevant violation observations. Specifically, for the predicate *P1=Happens(signal(R1,A1,S1),7,R(7,7)))* there is a single explanation Φ_{11}=*Happens(inspace(A1,S1),t1,R(2,7))* with basic probabilities $m_E(Valid(\Phi_{11}))\}=0.5$ and $m_E\ (\neg Valid(\Phi_{11}))\}=0$, as we discussed earlier. Thus, $Bel(Gen(P1))=m_E\ (Valid(\Phi_{11}))\}=0.5$ and $Bel(\neg Gen(P1))=m_E(\neg Valid(\Phi_{11}))\}=0$. The predicates *P2=HoldsAt(covers(R1,S1),7)* and *P3=HoldsAt(covers(R2,S1),7)* are also confirmed without using belief measures, as they are both derived from the runtime events (E1) and (E2) in Figure 2. Finally, *P4= ¬Happens(signal(R2,A1,S1),t,R(7,12))* is a negated predicate and, since no explanation of it can be generated from the assumptions of ATMS, the diagnosis process generates explanations of its positive form, i.e., *Happens(signal(R2,A1,S1),t,R(7,12))*. Following the same reasoning process as in the case of *P1*, Φ_{41}=*Happens(inspace(A2,S1,t,R(7,17))* will be derived as an explanation of *¬P4* with basic probabilities $mE(Valid(\Phi_{41}))\} = 0.5$ and $m_E\ (\neg Valid(\Phi_{41}))\} = 0$. Thus, *Bel(Gen(¬P4))=0.5* and *Bel(¬Gen(¬P4))= 0* and, from (F4) and (F5), *Bel(¬Gen(P4))=0.5* and *Bel(Gen(P4))= 0*. Thus, *P4* is reported as an unconfirmed predicate and, finally, as the cause of the rule violation.

4 Related work

In the context of model-based diagnosis, diagnosis focuses on the detection of system failures and typically involves the identification of traces of system events that have led to a failure (problematic event) using automata that recognise faulty behaviour [1][5][8][12][18]. In [5], diagnosis is carried through the synchronization of automata modelling the expected behaviour of a monitored system and the events captured from it. [8] has a similar but decentralised approach where synchronisation is performed for individual system components and then aggregated for the global system. In [1][18], the problem of fault diagnosis, concerning time, has been studied by using timed automata to model systems.

Our approach is different from the above, as our focus is not the detection of the cause of faulty behaviours (this is the subject of earlier work described in [16]) but the explanation of such causes in the presence of incomplete and/or not trusted event traces. Another difference between the work in model based diagnosis and our abduction based explanation process is that our process is based on Event Calculus for modelling not the whole system but only the properties, which should be monitored, and assumptions that could provide information related to the monitored properties.

The generation of abductive explanations considering temporal information is the main focus of interest of the research work described in [2] and [13]. In [2], a temporal abduction algorithm is described which makes use of temporal constraints associated with the observations and the formulation of the underlying domain theory. In [13], the time ranges of the generated explanations are calculated by the use of a computation method based on linear constraint satisfaction, while uncertainty of the explanations is treated by the use of probabilistic assessment scheme based on Bayesian inference [6].

Our approach as well draws upon work on temporal abductive reasoning [2][3][10][15] and its applications to diagnosis [3][9], but is based on a newly developed algorithm for abductive search with EC that generates all the possible alternative explanations of a formula (unlike [2][15]), treats the time constraint satisfaction problem as a linear programming problem and computes beliefs in explanations using the DS theory. These beliefs are also used in order to rank explanations and select some of them as the most plausible. The choice of the DS theory of evidence as the framework for calculating the likelihoods of abduced explanations has been dictated by the need to represent the uncertainty regarding the confirmation of the consequences of these explanations as we discussed in Section 3.3 and reason in the presence of this uncertainty. Also, by using the DS theory, we avoid the need to elicit the a-priori and conditional probability measures which are required by Bayesian inference [6].

5 Conclusions

In this paper, we have presented the extension of a framework supporting the runtime monitoring of software systems which can provide diagnostic information for violations of monitored properties. The provision of diagnostic information is based on alternative *explanations* of events involved in violations of properties which are generated by abductive reasoning using a model of the monitored properties expressed in Event Calculus. Our approach supports also the computation of beliefs in the plausibility of explanations based on evidence about their expected effects that is gathered from the event log of the monitored system. A more detailed account of our approach and its implementation is given in [17]. Currently, we are conducting an experimental evaluation of it in the context of industrial case studies of the SERENITY project.

References

1. Bouyer P., Chevalier F. and D'Souza D.: “Fault Diagnosis using Timed Automata”. In Proc. 8th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'05), LNCS 3441, 219--233, Springer (2005)
2. Console L. et al.: Local Reasoning and Knowledge Compilation for Efficient Temporal Abduction. IEEE Trans. on Knowledge & Data Engineering 14(6): 1230--1248 (2002)
3. De Kleer J., Williams B.C.: Diagnosing Mulitple Faults. Artif. Intell. 32(1): 97--130 (1987) Denecker M. et al.: Temporal reasoning with abductive event calculus. 10th ECAI (1992)
4. Gale D.: “Linear programming and the simplex method”. Notices of the AMS, 54(3):364--369 (2007)
5. Grastien A., Cordier M., Largouët C.: Incremental Diagnosis of Discrete-Event Systems, 15th Int. Work. On Principles of Diagnosis (DX05) (2005)
6. Heckerman, D.: A tutorial on learning with Bayesian networks. In Learning in Graphical Models, M. I. Jordan, Ed. MIT Press, Cambridge, MA, 301—354 (1999)
7. Lazarevic A., Kumar V., Srivastava J.: Intrusion detection: a survey. In Managing cyber-threats: issues approaches & challenges, Springer (2005)
8. Pencolé Y., Cordier M.: A formal framework for the decentralised diagnosis of large scale discrete event systems & its application to telecommunication networks. Artif. Intell. 164: 121—180 (2005)
9. Poole D.: Explanation and prediction: An architecture for default and abductive reasoning. Comp. Intell. 5(2): 97--110 (1989)
10. Ray O., Kakas A.: ProLogICA: a practical system for Abductive Logic Programming. 11th Int. Works. on Non-monotonic Reasoning, 304—312 (2006)
11. Reiter R.: A theory of diagnosis from first principles. Artif. Intell. 32(1): 57--96 (1987)
12. Sampath M et al.: Failure diagnosis using discrete-event models. IEEE Trans. on Control Systems Technology, 4(2):105--124 (1996)
13. Santos E. Jr.: “Unifying time and uncertainty for diagnosis”. Journal of Experimental and Theoretical Artificial Intelligence, 8, 75—94 (1996)
14. Shafer G.: A Mathematical Theory of Evidence. Princeton University Press (1975)
15. Shanahan M.: Abductive Event Calculus Planner. J. Logic Programming 44: 207--239 (2000)
16. Spanoudakis G., Mahbub K.: Non intrusive monitoring of service based systems. Int. J. of Cooperative Infor. Systems, 15(3):325--358 (2006)
17. Spanoudakis G., Tsigkritis T.: v1 of diagnosis prototype. Deliverable A4.D5.1, SERENITY Project, http://www.serenity-forum.org/ (2008)
18. Tripakis S.: Fault diagnosis for timed automata. In Proc. 7th Int. Symp. Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT’02), vol. 2469 of LNCS, 205--224, Springer (2002)