Chapter 14
Diagnosis and Threat Detection Capabilities of the SERENITY Monitoring Framework

Theocharis Tsigkritis, George Spanoudakis, Christos Kloukinas and Davide Lorenzoli

Abstract In addition to the basic monitoring capabilities that have been described in Chapter 13, the SERENITY monitoring framework offers mechanisms for diagnosing the reasons that have caused the violation of security and dependability (S&D) properties and detecting potential violations of such properties, called "threats". Diagnostic information and threat detection are often necessary for deciding what would be an appropriate reaction to a violation and taking preemptive actions that could stop the predicted violation, respectively. In this chapter, we describe the mechanisms of the SERENITY monitoring framework which are used to generate diagnostic information for violations of monitoring rules that express S&D properties within S&D Patterns, and predict potential violations of such properties.

14.1 Introduction

Monitoring security and dependability (S&D) properties during the operation of software systems is widely accepted as a measure of runtime verification that in-

Theocharis Tsigkritis
Dept. of Computing, City University,
Northampton Square, London, EC1V 0HB, e-mail: t7t@soi.city.ac.uk

George Spanoudakis
Dept. of Computing, City University,
Northampton Square, London, EC1V 0HB, e-mail: G.Spanoudakis@soi.city.ac.uk

Christos Kloukinas
Dept. of Computing, City University,
Northampton Square, London, EC1V 0HB, e-mail: c.kloukinas@soi.city.ac.uk

Davide Lorenzoli
Dept. of Computing, City University,
Northampton Square, London, EC1V 0HB, e-mail: Davide.Lorenzoli.1@soi.city.ac.uk

creases system resilience to dependability failures and security attacks. SERENITY advocates the need for this form of system verification and has developed a monitoring framework, called EVEREST, to support the monitoring of S&D properties during the operation of a system. The core capabilities of EVEREST have been described in Chapter 13. It should be noted, however, that whilst monitoring is able to detect violations of S&D properties at runtime, it cannot always provide information that is necessary for understanding the reasons that underpin the violation of an S&D property and making decisions about what would be an appropriate reaction to it. Furthermore, it is often necessary to try to predict the possibility of a violation using information about the current state of a system rather than wait until all the information that would enable a definite decision about the violation becomes available. This is because an accurate early prediction can widen the scope of possible reactions to the violation or even provide scope for taking pre-emptive action that prevents the violation.

To appreciate the need for diagnosing the reasons underpinning the violation of an S&D property, consider the location based access control system (LBACS) that was introduced in Chapter 13. As discussed in that chapter, LBACS grants access to different resources of an enterprise (e.g. printers, intranet, and internet) from mobile devices depending on the credentials of these devices and their exact location within the physical space of the enterprise. The location of a device in LBACS is determined by the strength of signals sent from the device to a location server. To ensure the availability of accurate information about the location of mobile devices in LBACS, each device is expected to send signals to the location detection server periodically.

In EVEREST, the absence of a signal after the elapse of a given signalling period can be detected by specifying a monitoring rule, requiring that the time between two consecutive signals from the same device should not exceed the given period. Detecting, however, the occurrence of a violation of this rule is not in itself sufficient for establishing the reasons why some device has failed to send the expected signals. In such cases, a further search for possible causes of the violation could be useful for deciding how to react to the violation. To appreciate why, consider that the violation might, for example, have been caused because: (a) the involved device malfunctions and has stopped sending signals after some time point, (b) the involved device is no longer present in the area covered by the server, (c) some of the signals sent by the device have been lost in the communication channel between the device and the server, or (d) the signal that was used to determine the start of the last period of checking was sent by an external agent (attacker) who managed to fake the identity of the device (i.e., an attacker). Although the above list of possible causes is not exhaustive, it demonstrates that a decision about what would be an appropriate reaction to the violation depends on the reason(s) that have caused it and, therefore, the selection of the appropriate response action cannot be made solely on the basis of knowledge about the violation but requires additional diagnostic information.

In this chapter, we present the mechanisms of EVEREST that support the generation of diagnostic information for detected violations of S&D properties and the prediction of potential violations of S&D properties.

The overall aim of the diagnostic mechanism of EVEREST is the identification of possible *explanations* for the violations of rules that have been detected by the framework in order to aid the selection of appropriate reactions to these violations. To generate such explanations, the diagnostic mechanism used *abductive reasoning* [22]. Then, following the identification of possible explanations, the diagnosis mechanism also assesses the plausibility of explanations by identifying any effects that they would have beyond the events that they were generated from and checking whether these effects correspond to events that have been recorded in the event log of the monitoring framework and are genuine. The assessment of the genuineness of the explanation effects and the validity of explanations is based on the computation of beliefs using functions that we have defined for this purpose. These functions have been defined using the axiomatic framework of the Dempster Shafer theory of evidence [18]. The diagnostic mechanism of EVEREST has been tested successfully using some of the industrial scenarios of the project. Examples of the application of this mechanism in the industrial scenarios of SERENITY and its results are discussed in this chapter.

The detection of potential violations of S&D properties in EVEREST builds upon the basic monitoring and diagnostic capabilities of the framework and is based upon the computation of beliefs that violations of such properties are likely to occur. The computation of such beliefs is based upon the diagnostic mechanisms of EVEREST which provide the basic assessment of the genuineness of the events received by the framework and historical data about the frequency of co-occurrence of events which are connected by temporal constraints within specific S&D monitoring rules. These historical data provide the basis for computing beliefs in the potential occurrence or not of an event when another event that it is constrained by has occurred and is known to be genuine.

The rest of this chapter is structured as follows. Section 14.2 provides an overview of the diagnostic process of the SERENITY monitoring framework. Sections 14.3, 14.4 and 14.5 focus on the processes of generating explanations, identifying their expected effects and assessing the validity of explanations and genuineness of events, respectively. Section 14.6 presents the threat detection mechanisms of the SERENITY monitoring framework and gives examples of threat detection. Finally, Section 14.7 gives an overview of related work and Section 14.8 provides some overall concluding remarks and outlines directions for future work.

14.2 Overview of Monitoring and the Diagnosis Process

As discussed in Chapter 13, the monitoring framework of SERENITY supports the monitoring of S&D properties during the operation of distributed systems which may bind (and re-bind) software components running on different devices dynamically. S&D properties are specified by a special form of *Event Calculus* [19] formulas, called *monitoring rules*. These formulas are checked against streams of runtime events which the SERENITY monitoring framework receives from the different components of the system which is being monitored, and/or information about the state of this system. The latter information is deduced using a set of *assumptions* about the behaviour of the system and how it affects its state.

A monitoring rule has the form $B_1 \wedge \ldots \wedge B_n \Rightarrow H$ where B_i and H can be predicates of one of the following two types:

- *Happens(e,t,R[t1,t2])* – this predicate indicates that an event *e* of instantaneous duration has occurred at some time point *t* within a given time range *[t1,t2].*
- *HoldsAt(f,t)* – this predicate indicates that a condition *f*, called fluent, is valid at some time point t.

The syntactic form and meaning of the terms *e* and *f* which are used to represent events and fluents in the above predicates have been discussed in chapter D3 and, therefore, are not discussed in more detail here.

A rule that could be specified in Event Calculus to represent the condition about the periodic dispatch (every *m* time units) of signals from the mobile devices in the scenario presented in Section 13.2 within the location based access control S&D Pattern that has been specified to support this scenario is the following:

```
Rule 1:

Happens(e(_e1, _devID, _locServerID, REQ, signal(_devID),
_locServerID), t1, R(t1,t1))
⇒(∃t2:Time,e2:String)
 Happens(e(_e2,_ devID, _locServerID,  REQ, signal(_devID),
_locServerID), t2, R(t1,t1+2)) ∧ (_e1 ≠ _e2)
```

The above rule would be violated by the following events that are stored in the log:

```
(E1)   Happens(e(E1, Lap33, LocSer1,REQ,signal(Lap33), LocSer1),15, R(15,15))
                                                        [event captor-LocSer1]
(E2)   Happens(e(E2, Lap33, AcConSer1,REQ,accessTo(Lap33,PrinterA1),
       AcConSer1),2,R(2,2))                          [event captor-AcConSer1]
(E3)   Happens(e(E3, Lap33, LocSer1,REQ,signal(Lap33), LocSer1),22, R(22,22))
                                                        [event captor-LocSer1]
```

More specifically, when the event E1 (i.e., *Happens(e(E1,Lap33, LocSer1,REQ,signal(Lap33),LocSer1),15, R(15,15)))* which indicates the receipt of a signal from *Lap33* by the location server *LocSer1* occurs, the rule would be satisfied only if within 2 time units from the receipt of the event, *LocSer1* had received a second signal from the same device. Note, however, that such a signal is not received as shown in the log. This is because following the receipt of the signal represented by the event E1, the next signal from *Lap33* is received at the time point t=22. Hence, the monitor can deduce by applying the principle of negation as failure (NF) the absence of a signal from *Lap33* from T=16 to T=17 or, formally, that the predicate *¬Happens(e(E1,Lap33, LocSer1,REQ,signal(Lap33), LocSer1),t, R(16,17)))* is *True*. It should be noted, however, that this deduction is possible only when the monitor receives event E3 in the log. This is because E3 indicates that the time of the location server is T=22 and, therefore, the monitor can deduce with certainty that it had received no other event from *Lap33* since receiving event E1 at T=15.

The diagnosis mechanism of SERENITY is invoked after the detection of a violation of a monitoring rule as the above in order to find possible explanations of the reasons underpinning the occurrence of the events involved in the violation the rule and assess their genuineness. This mechanism produces diagnostic information through a process of four stages, which are shown in Figure 14.1. These stages are:

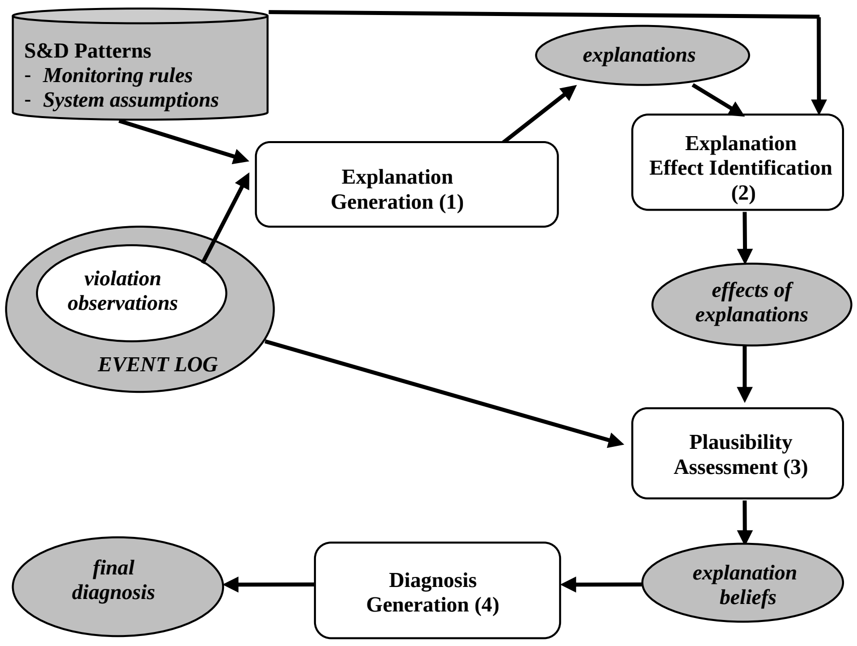


Fig. 14.1 The diagnosis process

1. The *explanation generation* stage – In this stage the diagnosis mechanism generates all the possible explanations of the events which are involved in the violation. These explanations are generated using abductive reasoning based on assumptions about the behaviour of the components of the system. These assumptions are specified as part of S&D Patterns.
2. The *explanation effect identification* stage – In this stage the diagnosis mechanism derives the possible consequences (effects) of the potential explanations that were identified in the previous stage. The consequences are generated from the abduced explanations and system assumptions using deductive reasoning.
3. The *plausibility assessment* stage – In this stage the diagnosis mechanism checks the expected effects of explanations against the event log to see if there are events that match them or, equivalently the existence of further supportive evidence for the explanation.
4. The *diagnosis generation* stage – In this stage the diagnosis mechanism produces an overall diagnosis for the violation including belief measures in the genuineness of the events involved in the violation and the most plausible explanations that have been identified for these events (if any).

In the following we describe each of the above stages in detail.

14.3 Generation of Explanations

The generation of explanations 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, the diagnosis mechanism searches for *explanation formulas* Φ which, in conjunction with the set of the *assumptions* about the system that is being monitored and the events recorded in the log of the monitor, entail the events in Ω.

The search for explanations starts from a violation observation *P* that needs to be explained and find all assumptions of the form a*:* $B_1 \wedge \ldots \wedge B_n \Rightarrow H$ whose head *H* can be unified with *P*. When such an assumption is found, the algorithm checks whether: (a) the unification of *P* with *H* provides concrete values for all the non time variables of the predicates $B_1, \ldots, B_n$ in its body, and (b) it is possible to derive concrete time ranges for the time ranges of all these predicates. If these conditions are satisfied, the algorithm instantiates the predicates $B_1, \ldots, B_n$ and identifies which of these predicates are observable predicates (*O-preds*), deducible predicates (*D-preds*) or abducible predicates (*A-preds*). The set of the generated explanations of an event E_i must be a subset of *A-preds.*

Then, the algorithm checks if each of the observable or deducible predicates in the body of *a* can be matched with some recorded event or can be derived from the events in the monitor's log and the known system assumptions, respectively. If there are observable or deducible that cannot be verified via this check, the algorithm tries to find abduced explanations for them recursively. If such explanations are found for all the non verified observable or deducible predicates, these explanations along with the abduced predicates 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 observables and/or deducible predicates 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 the explanation generation process consider the violation of *Rule 1* above. When presented with the events involved in this violation, namely the events represented by the predicates:

P1:Happens(e(E1,Lap33,LocSer1,REQ,signal(Lap33),LocSer1), 15, R(15,15)) and

P2:¬Happens(e(E1, Lap33, LocSer1, REQ, signal(Lap33), LocSer1),t, R(16,17))

the diagnosis mechanism tries to find possible explanations for each of these predicates individually. The search for such explanations is based on assumptions specified about the behaviour of the different components involved in the location based access control S&D Pattern. Two of the assumptions of this pattern are:

```
Assumption 1:
  Happens(e(_eID1, _x1,  _x2, _x3, _x4, _x5, _x6,
  operableInPremises(_deviceID), _x7, _x8), t1, R(t1,t1))
  ⇒(∃t2:Time)
  Happens(e(_eID2,_deviceID,_locationServerID, RES, si-
  gnal(_deviceID),_locationServerID), t2, R(t1,t1+2))

Assumption 2:
  Happens(e((_eID1,_deviceID,_controlServerID,REQ,accessTo(_deviceID,
  _resourceID),_controlServerID), t1, R(t1,t1))
  ⇒(∃t2:Time)
  Happens(_e(eID3,_deviceID,_locationServerID,
  RES,signal(_deviceID),_locationServerID), t2, R(t1-2, t1+2))
```

The above assumptions are specified in the location based access control S&D Pattern to express the circumstances under which mobile devices are expected to send signals. In particular, the former assumption (*Assumption 1*) states that if a device (*_deviceID*) becomes operational in the area that is controlled by the access control system at some time point (), it must send a signal to the location server within the expected signalling period (i.e., within 2 seconds). The predicate *Happens(e(_eID1, _x1, _x2, _x3, operableInPremises(_deviceID), _x4), t1, R(t1,t1))* in Assumption 1 represents the event that the device becomes operational in the space of the location based access control system and the predicate *Happens(e(_eID2, _deviceID, _locationServerID, RES, signal(_deviceID), _locationServerID), t2, R(t1, t1+2))*) expresses the event of dispatching a signal. The second assumption (*Assumption 2*) states that if a device requests the access control server to obtain access to a resource (see the predicate *Happens(e((_eID1,_deviceID,_controlServerID, REQ, accessTo(_deviceID, _resourceID),_controlServerID), t1, R(t1,t1)))* at some time point *t1*, then a signal must have been sent by this device in the period from 2 seconds prior to the resource access request to 2 seconds after it. The time period that is specified in the assumption for the dispatch of the device signal covers the possibility of the device having dispatched a signal prior to the request and the possibility of dispatching a signal after the request.

Given the above assumptions, the search for an explanation of the predicate *P1* above detects that this predicate can be unified with the predicate *Happens(e(_eID2, _deviceID, _locationServerID, RES, signal(_deviceID), _locationServerID), t2, R(t1-2, t1))* in the head of *Assumption 1* (the unifier of the two predicates is {*_eID2/E1, deviceID/Lap33, _locationServerID/LocSer1, t2/15*}).

Thus, the linear constraint system that is generated for the time variable t_1 in *Assumption 1* following the unification includes the constraints $t_1 \leq 15$ and $15 \leq t_1 + 2$ or, equivalently, $15–2 \leq t_1$ and $15 \leq t_1$. Hence, a feasible time range exists for t_1 (i.e., $t_1 \in [13,15]$) and, as the non time variables in the body of *Assumption 1* are covered by the unification , the conditions of the explanation generation process are satisfied and the predicate

Φ_{11}: Happens(e(_eID1, _x1, _x2, _x3, operableInPremises(Lap33), _x4), t1, R(13,15))

is generated as a possible explanation of *P1*. Intuitively, the meaning of this explanation is that as a signal has been received from the device *Lap33* at the time point *t=15*, it may be assumed that this device has become operational in the area covered by the access control system at some time point in the range from t=13 to t=15. Note also that since the predicate *Happens(e(_eID1, _x1, _x2, _x3, operableInPremises(Lap33), _x4), t1, R(13,15))* belongs to the set of abducible predicates, the explanation generation process does not attempt to produce more refined explanations of it.

Similarly, the predicate

Φ_{12}: *Happens(e((_eID1, Lap33, _controlServerID, REQ, accessTo(Lap33,_resourceID), _controlServerID), 15, R(13,17))*

can be generated as another possible explanation of *P1* from *Assumption 2* and assuming that Φ_{12} is also an abducible predicate. Hence, the set of possible explanations of the violation observation expressed by the predicate *P1*under the assumptions specified in the location based access control pattern includes the predicates Φ_{11} and Φ_{12}.

14.3 Identification of Explanation Effects

After the generation of the possible explanations for the events involved in the violation of a rule, the diagnosis process identifies the expected effects of these explanations and uses them to assess the plausibility of the explanations. The assessment of explanation plausibility is based on the hypothesis that if the expected effects of an explanation match with events that have occurred (and recorded) during the operation of the system that is being monitored, then there is evidence about the validity of the explanation. This is because the recorded events that match the expected effects of the explanation may have also been caused by the explanation itself. It should be noted that under the same hypothesis the violation observation (event) that the explanation was generated for also casts positive evidence for the explanation. However, the evidence that arises from this event is disregarded to avoid cycles in the reasoning process.

The identification of the expected effects of explanations is based on deductive reasoning. More specifically, given an explanation $Exp = P_1 \wedge \ldots \wedge P_n$ that is expressed as a conjunction of abduced predicates, the diagnosis process iterates over its constituent predicates P_i and, for each of them, it finds the system assumptions $B_1 \wedge \ldots \wedge B_n \Rightarrow H$ that have a predicate B_j in their body which can be unified with P_i and the rest of the predicates B_u (u=1,…,n and u≠j) in it are *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 expected 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*.

To clarify this stage of the diagnosis process, consider again the example of the violation of *Rule 1* in the location based access control pattern. Using *Assumption 1*, it is possible to determine the expected effects of the predicate that was generated as a possible explanation of *P1*. More specifically, from *Assumption 1* it is possible to identify the predicate

C_{111}: *Happens(e(_eID2, Lap33, LocSer1, REQ, signal(Lap33), LocSer1), t2, R(13,17))*

as an expected effect of Φ_{11}. Thus, according to this predicate any signal that has been sent by the device *Lap33* within the time range from *t=13* to *t=17* other than the signal represented by the predicate *P1* as we discussed above would support the validity of the explanation Φ_{11}.

Similarly, the expected effects of the other explanation of *P1*, i.e., the explanation Φ_{12} can be deduced from *Assumption 2*. Using this assumption, it can be deduced that the predicate

C_{121}: *Happens(e(_eID2, Lap33, _locationServerID, REQ, signal(Lap33), _locationServerID,), t2, R(11,19))*

is an expected effect of Φ_{12}. In the case of Φ_{12}, however, as the predicate *Happens(e((_eID1, Lap33, _controlServerID, REQ, accessTo(Lap33,_resourceID), _controlServerID), 15, R(13,17))* is also an observable predicate, it can also be regarded as a consequence of itself that should be verified against the event log of the monitoring framework. Hence, the set of the expected effects of Φ_{12} will be: $\Phi_{12}^{C} = \{\Phi_{12}, C_{121}\}$.

14.4 Assessment of Event Genuineness

14.4.1 Foundations of Assessment

The diagnosis mechanism assesses the genuineness of violation observations and other events based on the validity of the explanations that have been found for them. The basic principles that underpin this assessment are that:

1. an event is genuine if it has occurred and has at least one valid explanation, and
2. an explanation is valid for an event if its preconditions and expected consequences match with events in the log of the monitoring framework which are genuine themselves.

Based on the above principles, event genuineness is defined as follows:

Definition 1: The genuineness of an event E_i given a set of previous explanations E_o and a time range of interest $[T_{min}, T_{max}]$ is defined as:

$\text{Genuine}(E_i,[T_{min},T_{max}], E_o) = \text{Occurred}(E_i) \wedge \text{Explainable}(E_i, [T_{min},T_{max}],E_o)$
where

- *Occurred(E_i)* is a proposition denoting that the event E_i has occurred. This proposition is defined as:

  $$\text{Occurred}(E_i) = \text{True} \quad \text{if} \quad ((\exists E_w.\ E_w \in \text{EventLog}\,[T_{min},T_{max}]) \wedge \text{mgu}(E_i,E_w) \neq \emptyset \wedge (t_i^{LB} \leq t_w \leq t_i^{UB}))$$
  $$\text{Occurred}(E_i) = \text{False} \quad \text{Otherwise}$$

  where
  - *EventLog[T_{min},T_{max}]* is a subset of the events in the log of the monitor that includes only events which have occurred in the time range [T_{min},T_{max}]
  - *mgu(E_i,E_w)* is the most general unifier of the events E_i and E_w [9]
  - t_i^{LB} and t_i^{UB} are the lower and upper boundary of the time range within which E_i has occurred (or is expected to occur)[1]

- *Explainable(E_i, [T_{min}, T_{max}], E_o)* is a proposition denoting that the event E_i has at least one valid explanation in the time range *[T_{min}, T_{max}]* that is formally defined as:

  $$\text{Explainable}(E_i, [T_{min},T_{max}],E_o) = \vee_{\Phi ij \in EXP}\text{Valid}(\Phi_{ij},[T_{min},T_{max}],E_o \cup \{\Phi_{ij}\})$$

  where
  - *EXP_i* is the set of the alternative explanations that can be generated for the event E_i excluding any explanations already in E_o
  - *Valid(Φ_{ij},[T_{min}, T_{max}],E)* is a proposition denoting that the explanation Φ_{ij} is valid within the time range of interest [T_{min},T_{max}] This proposition is defined as:

    $$\text{Valid}(\Phi_{ij}, [T_{min}, T_{max}], E) = \wedge_{E_u \in \text{Conds}(Ei,\Phi ij,\ [Tmin,\ Tmax])} \text{Genuine}(E_u, [T_{min},T_{max}], E)\} \wedge \{\vee_{E_w \in \text{Cons}\,(Ei,\Phi ij,\ [Tmin,\ Tmax])} \text{Genuine}(E_w, [T_{min}, T_{max}], E)\})$$

  - *Conds(E_i,Φ_{ij},[T_{min},T_{max}])* is the set of the preconditions in the deductive path from Φ_{ij} to E_i, that are expected to occur within the diagnosis range [T_{min},T_{max}][2]. This set is defined as:

    $$\text{Cond}\,(E_i,\Phi_{ij},[T_{min},T_{max}]) = \{E_w |\ E_w \in \text{EventLog}[T_{min},T_{max}] \wedge \exists c,f.\ (f \in \text{DeductivePath}(\Phi_{ij},E_i)) \wedge (c \in \text{Body}(f)) \wedge \text{mgu}(c,E_w) \neq \emptyset\ \}$$

  - *Cons (E_i,Φ_{ij},[T_{min},T_{max}])* is the set of the consequences of the explanation Φ_{ij} that are expected to occur within the diagnosis range [T_{min},T_{max}], defined as:

    $$\text{Cons}(E_i,\Phi_{ij},[T_{min},T_{max}]) = \{E_k \mid (\Phi_{ij} \vdash E_k) \wedge E_k \in \text{EventLog}[T_{min},T_{max}]\ \}$$

[1] t_i^{LB} and t_i^{UB} are both equal to the timestamp t_i of E_i if E_i is an event in the log of the monitor.

[2] This set is determined during the abductive reasoning process that generates the explanation Φ_{ij}.

According to definition 1, the occurrence or not of an event is assessed by checking whether or not the event has been recorded in the log of the monitor. Although this criterion provides only an indirect indication of the occurrence or not of an event and in some cases may lead to uncertainty about the non occurrence of events as we discuss below, it provides the only basis for making similar decisions at runtime. It should also be noted that Definition 1 restricts the assessment of event genuineness within a time range *$[T_{min},T_{max}]$*, called "diagnosis window". The diagnosis window is set by the user of the SERENITY runtime framework and determines the boundaries of the space within which the search for supportive evidence for events is performed. Imposing such boundaries is often necessary for the efficiency of the diagnosis process since without them the recursive definition of event genuineness can lead to an exhaustive search of the entire log of the monitor.

Another characteristic of Definition 1 is that during the recursive assessment of event genuineness, it excludes from the possible explanations of the event which is currently being assessed any explanations that are part of a set E_o. During diagnosis, the set E includes any explanations that have been generated earlier in for other events and which need to be excluded during the assessment of the current event in order to avoid circles in the assessment process. In the example shown in Figure 14.2, for instance, the assessment of the genuineness of the event e_3 will require the assessment of the genuineness of the event e_1 since the latter event is an expected consequence of the explanation Φ of e_3. Thus, according to Definition 1, the genuineness of event e_1 will need to be assessed as part of the process of assessing the genuineness of e_3. When assessing the genuineness of e_1, however, the explanation Φ will have already been generated as an explanation of e_3, and therefore it will be disregarded during the assessment of e_1.

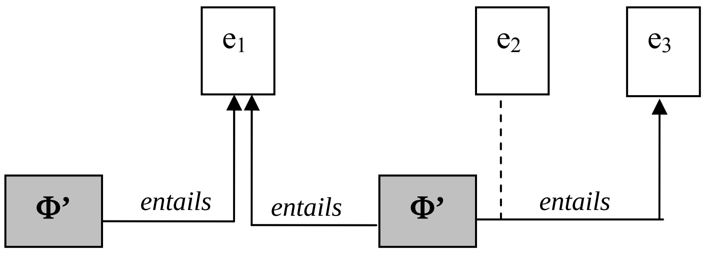


Fig. 14.2 Events and explanations

Definition 1 establishes the logical criteria for the assessment of event genuineness. It should be noted, however, that during diagnosis there might be an uncertainty about the occurrence or not of some events and in such cases the exact reasoning imposed by this definition would be over restrictive characterising most of the events as non genuine. More specifically, when searching the log of the monitor to find events that could match (and, therefore, confirm) an expected consequence or precondition of an explanation, as required by Definition 1, there is a possibility to fail to find a matching event not because this event has not occurred but because it has not arrived at the monitor yet. This can happen because the time that it takes for an event to arrive from the source where it has been captured to the monitor is not negligible and may vary depending on network traffic. This possibility can be excluded in some cases but not all. More specifically, since the communication channels between different event captors and the SERENITY runtime framework are assumed to operate in a FIFO mode (i.e., to send the events in the same order that they receive them[3]) when the diagnosis process searches for an event *e* that is expected to have occurred by some time point t_e at the latest and does not find it in the log, it can check the timestamp of the latest event that has been received from the event captor that is expected to send *e*. If this timestamp is greater than t_e then it can be guaranteed that *e* will not arrive at the monitor at a later time point and the absence of this event (i.e., $\neg e$) can be assumed to be true. If t_{last}, however, is less than t_e then it is impossible to establish with certainty whether or not *e* has occurred.

To deal with this uncertainty, the diagnosis mechanism of SERENITY advocates an approximate reasoning approach which generates degrees of belief in the membership of an event in the log of the monitor and the existence of some valid explanation for it rather than strict logical truth values. These degrees of belief are computed by functions founded in the axiomatic framework of the Dempster Shafer theory of evidence [18]. These functions are introduced in the following.

14.4.2 Belief Functions

The diagnosis mechanism of SERENITY uses a set of belief functions to generate beliefs in the explainability of events and whether or not they have occurred. The two main functions which are used for this purpose are:

1. the function m^{EX} which measures the basic degree belief that an event E is explainable (i.e., the basic degree of belief in the proposition denoted by Explainable(E, $[T_{min}, T_{max}]$, E_o)) , and
2. the function m^{IL} which measures the basic degree belief that an event E is in the log of the monitor (i.e., the basic degree of belief in the proposition denoted by Occurred(E))

These two functions are defined as follows:

Definition 2: m^{EX} is a function measuring the basic degree of belief in the existence of a valid explanation for an event E_i defined as:

$m^{EX}(\text{Explainable}(E_i, [T_{min}, T_{max}], E_o)) = 1$ if $E_i = P_{NULL}$
$m^{EX}(\text{Explainable}(E_i, [T_{min}, T_{max}], E_o)) = \alpha_1$ if $E_i = C_{NULL}$
$m^{EX}(\text{Explainable}(E_i, [T_{min}, T_{max}], E_o)) = \alpha_2$ if $E_i \neq P_{NULL}$, $E_i \neq C_{NULL}$, & $EXP = \emptyset$
$m^{EX}(\text{Explainable}(E_i, [T_{min}, T_{max}], E_o)) =$

[3] Network channels that realise the TCP/IP protocol have this property.

$\sum_{I \subseteq EXP \text{ and } I \neq \varnothing} (-1)^{|I|+1} \{\prod_{x \in I} m^{VL}(\text{Valid}(x, E_i,[T_{min},T_{max}],E_o \cup \{x\}))$ Otherwise

where

- *EXP* is the explanation set of E_i that has been defined in Definition 1
- *$m^{VL}(Valid(x,[T_{min},T_{max}],E_o \cup \{x\})$* is the basic probability in the validity of the explanation x of E_i that is computed according to Definition 3 below.

Definition 3: *m^{VL}* is a function measuring the basic probability of the validity of an explanation Φ for an event E, given a diagnosis window $[T_{min},T_{max}]$ and a set of prior explanations E_o, defined as:

$m^{VL}(\text{Valid}(\Phi, E, [T_{min},T_{max}], E_o) =$
$\prod_{E_u \in CND(E,\Phi,[Tmin,Tmax])} m^{GN}(\text{Genuine}(E_u,[T_{min},T_{max}], E_o \cup \{\Phi\})) \times$
$\{\sum_{S \subseteq CNS(E,\Phi,[Tmin,Tmax]) \,\&\, S \neq \varnothing} (-1)^{|S|+1} \{\prod_{E_w \in S} m^{GN}(\text{Genuine}(E_w, [T_{min}, T_{max}],$
$E_o \cup \{\Phi\}))\}\}$

where

- *EXP* is the explanation set of E that has been defined in Definition 1
- *$CND(E,\Phi,[T_{min},T_{max}])$* is a set that includes the preconditions in the deductive path from an explanation *Φ* to the event *E* that are expected to be satisfied within the diagnosis window $[T_{min},T_{max}]$. If there are no such preconditions, *$Conds(E,\Phi,[T_{min},T_{max}])$* includes a single special element *P_{NULL}* which denotes the absence of any precondition in the deductive path and is called *NULL* precondition. *$CND(E,\Phi,[T_{min},T_{max}])$* is defined as:

  $CND(E,\Phi,[T_{min},T_{max}]) = \{P_{NULL}\}$ If $Conds(E,\Phi,[T_{min},T_{max}])=\varnothing$
  $CND(E,\Phi,[T_{min},T_{max}]) = Conds(E_i,\Phi_{ij},[T_{min},T_{max}])$ Otherwise

- *$CNS(E,\Phi, [T_{min}, T_{max}])$* is the set of the consequences of the explanation Φ of E that are expected to occur within the diagnosis range $[T_{min},T_{max}]$. If no such consequences exist, *$CNS(E,\Phi,[T_{min},T_{max}])$* contains a special element *C_{NULL}* which denotes the absence of any consequence of Φ in the relevant time period and is called *null consequence*. *$CNS(E,\Phi,[T_{min},T_{max}])$* is defined as:

  $CNS(E,\Phi, [T_{min},T_{max}]) = \{C_{NULL}\}$ If $Cons(E_i,\Phi_{ij},[T_{min},T_{max}])=\varnothing$
  $CNS(E,\Phi, [T_{min},T_{max}]) = Cons(E_i,\Phi_{ij},[T_{min},T_{max}])$ Otherwise

- *m^{GN}* is a function that measures the basic probability of the genuineness and non-genuineness of an event and is defined by Definition 4.

Definition 4: *m^{GN}* is a function that measures the basic probability of the genuineness and non-genuineness of an event, defined as:
$m^{GN}(\text{Genuine}(E_i,[T_{min},T_{max}],E_o)) =$
$m^{IL}(\text{Occurred}(E_i)) \times m^{EX}(\text{Explainable}(E_i,[T_{min},T_{max}],E_o))$

$m^{GN}(\neg\text{Genuine}(E_i, [T_{min},T_{max}], E_o)) =$
$m^{IL}(\neg\text{Occurred}(E_i)) + m^{EX}(\neg\text{Explainable}(E_i, [T_{min},T_{max}], E_o)) -$

$(m^{IL}(\neg Occurred(E_i)) \times m^{EX}(\neg Explainable(E_i, [T_{min},T_{max}], E_o))$

As indicated in definitions 2 and 3, m^{EX} calculates the basic probability of the explainability of an event E_i as the belief in the existence of at least one valid explanation for it, i.e., an explanation whose preconditions for the entailment of E_i are all genuine and which has at least one genuine consequence other than E_i. Thus, the computation of m^{EX}_i is recursive.

It should also be noted that m^{EX} assigns some belief in the genuineness of events which have no explanations. This is a relaxation of the logical definition of event genuineness in Definition *1* that is introduced for two reasons. The first reason is that an event E_i with no explanations of its own may be required to provide confirmatory evidence for a precondition or consequence of an explanation of another event E_j. If this is the case, the assignment of a zero belief in the explainability of E_i (due to the absence of an explanation for it) would reduce to zero the basic probability of the genuineness of E_i. Thus, it could also reduce or even make equal to zero the basic probability of the genuineness of the event E_j whose explanation had to be confirmed by E_i. The stance reflected by definitions 2 and 3 in this case is that the very presence of E_i in the monitor's log should provide some evidence for the validity of the explanation of E_j even though E_i is not explainable itself and that the belief in the validity of this explanation should be higher than in cases where none of its consequences of preconditions were matching with events in the monitor's log. The second reason for assigning a non zero basic probability in the explainability of events with no explanations is because during diagnosis, an event may end up having no explanations either because its explanations have already been considered as explanations of previous events or because they fall outside the selected diagnosis window. In such cases, it would not be accurate to cast a zero belief in the genuineness of the event. To avoid such cases, m^{EX} assigns a small belief in the genuineness of events with no explanation that is determined by the parameter α_2. The value of this parameter should be set very close to zero, in order to provide a close approximation of the logical definition of explainability (Definition 1) in cases where an event does not have any explanation.

Finally, m^{EX} assigns a pre-determined basic probability to null preconditions and consequences (i.e., 1 and α_1, respectively). The former belief measure reflects the stance explanations with no preconditions are as valid as explanations with genuine preconditions. The assignment of the basic probability measure α_1 to an explanation with no consequences addresses a different consideration. More specifically, whilst the reasoning principle underpinning the diagnosis framework of SERENITY favours explanations which are confirmed by the fact that they have consequences matched by genuine events other than the event that they were generated for, it would be unfair to disregard entirely explanations that have no other such consequences. Cases of such explanations are more likely to arise when the diagnosis window is narrow and, therefore, it may be possible to end up with explanations with no further consequences falling within the given diagnosis window. For such explanations, it is important to assign some basic probability measure in their validity but at the same time keep this measure low to reflect the absence of any evidence of runtime event in the given diagnosis interval. The definition of the basic probability function m^{EX} introduces the parameter α_1 to define the belief measure that should be used in such cases and leaves the choice of its exact value to the user of the framework. The expectation, however, is that this value will be a number close to zero to ensure that explanations with no consequences cannot affect significantly the overall belief in the genuineness of events. Furthermore, it should be noted that α_1 must be greater than α_2 to ensure that explanations with no consequences affect more the beliefs in the genuineness of events than null explanations.

The basic probability of whether or not an event has occurred is computed by the function m^{IL}. This function is defined as follows:

Definition 5: m^{IL} is a function measuring the basic probability in the occurrence of an event E_i defined as:

$m^{IL}(\text{Occurred}(E_i)) = 1$	if $E_i \in$ EventLog or E_i is a *HoldsAt*, P_{NULL} or C_{NULL} predicate
$m^{IL}(\neg\text{Occurred}(E_i)) = 1$	if $E_i \notin$ EventLog, and E_i is not a *HoldsAt*, P_{NULL} or C_{NULL} predicate, and *lastTimestamp(event captor(E_i))* $\geq t_i^{UB}$
$m^{IL}(\text{Occurred}(E_i) \vee \neg\text{Occurred}(E_i)) = 1$	if $E_i \notin$ EventLog, E_i is not a *HoldsAt*, P_{NULL} or C_{NULL} predicate, and *lastTimestamp(event captor(E_i))* $< t_i^{UB}$

According to this definition, m^{IL} assigns a basic probability of 1 to *Occurred(E_i)* for any event E_i that has been recorded in the log of the monitor. This belief reflects the certainty about the occurrence of any event that has appeared in the monitor's log. m^{IL} assigns also a basic probability of 1 to *Occurred(C_{NULL})* and *Occurred(P_{NULL})* as these are special events denoting the absence of explanation consequences and preconditions for particular explanations and when they are established by the reasoning processes of the diagnosis framework they can be assumed to be always true[4]. m^{IL} also assigns a basic probability of 1 to *¬Occurred(E_i)* for any event E_i that has not been recorded in the log when m^{IL} is invoked and, at the time of the invocation, the timestamp of the latest event which has been generated by the event captor that is expected to produce E_i and recorded in the log (i.e., *lastTimestamp(event captor(E_i))*) is greater than or equal to the maximum time boundary until when E_i should have occurred (i.e., t_i^{UB}). When, however, at the time of the invocation of m^{IL}, the event in question (E_i) is not in the log but the timestamp of the latest event that has been received from the event captor that is expected to produce E_i is less than t_i^{UB}, m^{IL} assigns a basic probability of 1 to *Occurred(E_i) ∨ ¬ Occurred(E_i)* and a basic probability of 0 to each of *Occurred(E_i)* and *¬Occurred(E_i)* (i.e., $m^{IL}(\text{Occurred}(E_i)) = m^{IL}(\neg\text{Occurred}(E_i)) = 0$ and $m^{IL}(\text{Occurred}(E_i) \vee \neg\text{Occurred}(E_i))=1$). The assignment of these basic

[4] The need to provide an m^{IL} basic probability measure to such events may arise due to the recursive definition of m^{EX} in terms of m^{GN} and m^{IL}.

probabilities in the latter case represents the complete uncertainty about the occurrence or not of E_i within the time period that the event is expected to occur. This uncertainty is also the reason for selecting the *DS theory* (as opposed to the classic probability theory) as the axiomatic framework for the belief based assessment of event genuineness (if $m^{IL}(Occurred(E_i))$ and $m^{IL}(\neg Occurred(E_i))$ were classic probabilities it should always be that $m^{IL}(Occurred(E_i)) + m^{IL}(\neg Occurred(E_i))=1$ and therefore it would not be possible to represent the uncertainty that may arise in this case).

The functions m^{GN}, m^{EX}, m^{VL}, and m^{IL} satisfy the axioms that define *basic probability assignments* in the DS theory and are, therefore, interpreted as functions of this type. A basic probability assignment *m* in the DS theory is a function from the power set of a set θ of mutually exclusive propositions, called "frame of discernment", to the real interval [0,1] that satisfies two conditions:

(A1) $m(\varnothing) = 0$ and

(A2) $\Sigma_{P\subseteq\theta}\, m(P) = 1$

Basic probability assignments in the DS theory provide measures of belief in the truth of the disjunction of the propositions in different subsets of θ that cannot be split to any of these propositions individually. This, however, is not the only degree of belief that can be casted to a proposition in DS theory. From the basic probability assignment it is possible to derive two more degrees of belief for a proposition: its *combined belief* and its *plausibility*. These two beliefs are computed by the so called “belief functions”. A belief function is induced from a basic probability assignment according to the following formula (axiom):

(A3) $Bel(A) = \Sigma_{B\subseteq A}\, m(B)$

As expressed by axiom (A3), the belief function *Bel* that is induced from a basic probability assignment *m* measures the total belief that is committed to the set of propositions *P* by accumulating the basic probability measures which are committed to the different subsets of *P* by *m*. Also based on the *Bel* function it is also possible to compute the plausibility of a proposition using the formula:

(A4) $Pls(A) = 1 - Bel(\neg A)$

Based on (A3), the basic probability assignment to event genuineness m^{GN} induces a belief function Bel^{GN} that has the following functional form:

$Bel^{GN}(Genuine(E_i,[T_{min},T_{max}],E_o)) = m^{GN}(Genuine(E_i,[T_{min},T_{max}],E_o))$

$Bel^{GN}(\neg Genuine(E_i,[T_{min},T_{max}],E_o)) = m^{GN}(\neg Genuine(E_i,[T_{min},T_{max}],E_o))$

14.4.3 Example

Following the introduction of the basic probability functions m^{EX}, m^{VL}, m^{IL} and m^{GN} we can now revisit the example of the violation of *Rule 1* and explain the use of these functions in the computation of basic degrees of belief in the genuineness of the events involved in this violation.

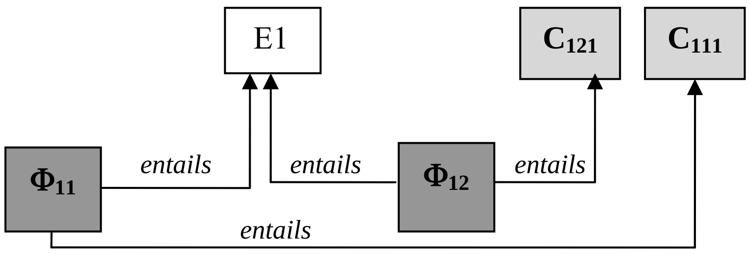
Recall that, as we discussed earlier in Section 14.3, the events that caused the violation of *Rule 1* were:

- *E1: Happens(e((E3, Lap33, LocSer1,REQ,signal(Lap33),LocSer1),15, R(15,15) and*
- *E4: ¬Happens(e((E3, Lap33, LocSer1,REQ,signal(Lap33), LocSer1),t,R(15,17))*

and the event that enabled the derivation of E4 (by application of the principle of negation-as-failure) was the event E3 in the log that occurred at T=22. Thus, assuming a required diagnosis period of 7 time units from the time point when the violation was detected, the boundaries of the diagnosis window will be T_{min}=15 and T_{max}=22.

Within this diagnosis window, the set of the alternative explanations that can be generated for *E1* includes the explanations Φ_{11} and Φ_{12}, as we discussed in Section 14.4. Also, their expected consequences are *Cons(E1,*Φ_{11}*,[15,22])*=\{C_{111}\} and *Cons(E1,*Φ_{12}*,[15,22])*=\{Φ_{12},C_{121}\}, as they are shown in Figure 14.3. Furthermore, it should be noted that the deductive paths from Φ_{11} and Φ_{12} to *E1* have no preconditions. Thus, *Conds(E1,*Φ_{11}*,[15,22])* = *Conds(E1,*Φ_{12}*,[15,22])* = ∅ and, by virtue of Definition 2, a null precondition will need to be considered in both of these cases or, equivalently, the evaluation of the basic probability in the explainability of *E1* will need to be based on the precondition sets *CND(E1,*Φ_{11}*,[15,22])* = *CND(E1,*Φ_{12}*,[15,22])*= \{P_{NULL}\}.

Based on the identification of these explanations and their consequences in the first and second phase of the diagnosis process, the third phase of the process starts by searching for confirmatory evidence for the identified consequences and preconditions of C_{111}, C_{121}, and Φ_{12} in the event log (events *E1-E3*). This search will detect that there are no recorded events that can be unified with C_{111} and C_{121} but there is an event (*E2*) that can be unified with Φ_{12}.



E1: **Happens**(e((E1,Lap33,LocSer1,REQ,signal(Lap33),LocSer1),15,R(15,15))
[event captor-LocSer1]

Φ_{11}: **Happens**(e(_eID1,_x1,_x2,_x3,operableInPremises(Lap33),_x4),t1,R(13,15))

Φ_{12}: **Happens**(e((_eID1,Lap33,_controlServerID,REQ,accessTo(Lap33,_resourceID), _controlServerID), 15, R(13,17))

C_{111}: **Happens**(e(_eID2, Lap33, LocSer1, REQ, signal(Lap33), LocSer1), t2, R(13,17))

C_{121}: **Happens**(e(_eID2, Lap33, LocSer1, REQ, signal(Lap33), LocSer1), t2, R(11,19)

Fig. 14.3 Possible explanations of event E1 and their consequences

Based on these results, the basic probability in the explainability of E1 will be:

$m^{EX}(\text{Explainable}(E1, [15, 22], \varnothing) =$
$\{\prod_{E_u \in CND(E1,\Phi_{11},[15,22])} m^{GN}(\text{Genuine}(E_u,[15,22], \{\Phi_{11}\})) \times$
$\{\sum_{S \subseteq CNS(E1,\Phi_{11},[15,22]) \text{ and } S \neq \varnothing} (-1)^{|S|+1} \{\prod_{E_w \in S} m^{GN}(\text{Genuine}(E_w,[15,22],\{\Phi_{11}\})\}\} +$
$\{\prod_{E_u \in CND(E1,\Phi_{12},[15,22])} m^{GN}(\text{Genuine}(E_u,[15,22], \{\Phi_{12}\})) \times$
$\{\sum_{S \subseteq CNS(E1,\Phi_{12},[15,22]) \text{ and } S \neq \varnothing} (-1)^{|S|+1} \{\prod_{E_w \in S} m^{GN}(\text{Genuine}(E_w,[15,22],\{\Phi_{12}\})\}\} -$
$\{\prod_{E_u \in CND(E1,\Phi_{11},[15,22])} m^{GN}(\text{Genuine}(E_u,[15,22], \{\Phi_{11}\})) \times$
$\{\sum_{S \subseteq CNS(E1,\Phi_{11},[15,22]) \text{ and } S \neq \varnothing} (-1)^{|S|+1} \{\prod_{E_w \in S} m^{GN}(\text{Genuine}(E_w,[15,22],\{\Phi_{11}\})\}\} \times$
$\{\prod_{E_u \in CND(E1,\Phi_{12},[15,22])} m^{GN}(\text{Genuine}(E_u,[15,22], \{\Phi_{12}\})) \times$
$\{\sum_{S \subseteq CNS(E1,\Phi_{12},[15,22]) \text{ and } S \neq \varnothing} (-1)^{|S|+1} \{\prod_{E_w \in S} m^{GN}(\text{Genuine}(E_w,[15,22],\{\Phi_{12}\})\}\}$ (1)

As, however, *$CND(E1,\Phi_{11},[15,22])=CND(E1,\Phi_{12},[15,22])=\{P_{NULL}\}$*, we will have:

$\prod_{E_u \in CND(E1,\Phi_{11},[15,22])} m^{GN}(\text{Genuine}(E_u,[15,22], \{\Phi_{11}\})) = 1$, and
$\prod_{E_u \in CND(E1,\Phi_{12},[15,22])} m^{GN}(\text{Genuine}(E_u,[15,22], \{\Phi_{12}\})) = 1$.

Thus, (1) will be equal to:

$m^{EX}(\text{Explainable}(E1, [15, 22], \varnothing) =$
$\{1 \times m^{GN}(\text{Genuine}(C_{111},[15,22],\{\Phi_{11}\})\} +$
$\{1\times\{m^{GN}(\text{Genuine}(\Phi_{12}, [15, 22], \{\Phi_{12}\})) + m^{GN}(\text{Genuine}(C_{121}, [15, 22], \{\Phi_{12}\})) -$
$m^{GN}(\text{Genuine}(\Phi_{12},[15, 22],\{\Phi_{12}\})) \times m^{GN}(\text{Genuine}(C_{121},[15, 22],\{\Phi_{12}\}))\} -$
$\{1\times m^{GN}(\text{Genuine}(C_{111},[15,22],\{\Phi_{11}\})\} \times$
$\{1\times\{m^{GN}(\text{Genuine}(\Phi_{12}, [15, 22], \{\Phi_{12}\})) + m^{GN}(\text{Genuine}(C_{121}, [15, 22], \{\Phi_{12}\})) -$
$m^{GN}(\text{Genuine}(\Phi_{12},[15, 22],\{\Phi_{12}\})) \times m^{GN}(\text{Genuine}(C_{121},[15, 22],\{\Phi_{12}\}))\}$ (2)

Note, however, as there are no recorded events that can be unified with C_{111} and C_{121} in the event log, we have:

$m^{GN}(\text{Genuine}(C_{111},[15,22],\{\Phi_{11}\})) = 0$
$m^{GN}(\text{Genuine}(C_{121},[15,22],\{\Phi_{12}\})) = 0$

Thus, (2) becomes:

$$m^{EX}(\text{Explainable}(E1, [15, 22], \varnothing) = m^{GN}(\text{Genuine}(\Phi_{12}, [15, 22], \{\Phi_{12}\}))) \qquad (3)$$

E2, however, can be unified with Φ_{12} and EXP $_{E2} = \{E_{NULL}\}$. Thus, we will have
$m^{GN}(\text{Genuine}(\Phi_{12},[15, 22],\{\Phi_{11}\})) = m^{IL}(\text{Occurred}(E2)) \times m^{EX}(\text{Explainable}(E2))$
$= \alpha_2$

Therefore, it will be $m^{EX}(\text{Explainable}(E1, [15, 22], \varnothing) = \alpha_2$

The disbelief in the genuineness of the event E1 is computed using the formula $Bel(\neg Genuine(E_i, [T_{min}, T_{max}],E_0)) = Bel(Genuine(\neg E_i, [T_{min}, T_{max}],E_0))$. As E1 exists in the event log there will be that Occurred(E1) = True and Occurred(¬E1) = False and, therefore, the belief in the genuineness of absence of E1 is zero ($Bel(\neg Genuine(E1, [15, 22],\varnothing)) = 0$)

14.5 Diagnosis Generation

The last phase of the diagnosis process is concerned with the generation of a final diagnosis of a violation based on the beliefs computed for the genuineness of the individual events involved in it. This final diagnosis is a report of the confirmed and unconfirmed predicates, which are involved in the violation that is generated as shown in the algorithm of Figure 14.4.

More specifically, this algorithm takes as input a template that represents an instantiation of an S&D monitoring rule that has been violated and generates explanations for the individual predicates which are involved in the violation by calling the newly developed *Explain* function initially (see lines 3 and 6 in Figure 14.4). In the case of negated predicates, the explanations are generated for the positive form of the predicate. This is because negated predicates cannot appear in the head of assumptions and, therefore, it is not possible to generate explanations for them directly. By virtue, however, of attempting to generate an explanation for the positive form of a negated predicate, the diagnosis process can still establish beliefs in the genuineness of the event represented by the predicate as we discussed above. It should also be noted that, as they do not appear in assumption heads, negated predicates cannot have been generated by deduction from assumptions during the monitoring process. Thus, their presence in violated rule instances is established by the principle of negation as failure when the expected predicate has not been seen in the event log of the monitoring system within the time range that it is expected to occur. Thus, an attempt to generate an explanation for the positive form of the predicate during the diagnosis process provides a means of confirming or not whether the application of the principle of negation as failure was reasonable given evidence from other events in the event log.

```
Generate_Violation_Explanation(R: Instance of Violated Rule)
1.  For each predicate P in R Do
2.    If P is negated Then
3.        Explanations(P) = explain(¬P, $t_{min}$(P), $t_{max}$(P), NULL)
4.        Generate_AE_Consequences(Explanations(P), Assumptions, P_Consequences)
5.    Else
6.        Explanations(P) = explain(P, $t_{min}$(P), $t_{max}$(P), NULL)
7.        Generate_AE_Consequences(Explanations(P), Assumptions, P_Consequences)
8.  End If
9.  [Bel(P), …, Pls(P)] = ComputeBeliefRange(P, Explanations(P), P_Consequences)
10.   If 1-Pls(P) < Bel(P) Then
11.       If P is negated Then
12.           UnconfirmedPredicates = UnconfirmedPredicates ∪ {P}
13.       Else
14.           ConfirmedPredicates = ConfirmedPredicates ∪ {P}
15.       End if
16.   End if
17. End For
18. For all P in ConfirmedPredicates Do report P as a confirmed predicate End for
19. For all P in UnconfirmedPredicares Do report P as unconfirmed predicate End for
END
```

Fig. 14.4 Explanation generation algorithm

Having generated explanations for the individual predicates, the *Generate_Violation_Explanation* algorithm computes a belief range for the event represented by each predicate and classifies the predicate as confirmed or unconfirmed depending on whether the belief in the genuineness of the event represented by it exceeds the belief in the non genuineness of this event. More specifically, a non negated predicate P will be classified as a confirmed predicate if Bel(P) > Bel(¬P)[5]. A negated predicate ¬P, will be classified as a unconfirmed predicate if Bel(P) > Bel(¬P). Finally, the algorithm reports the classifications of individual predicates as confirmed or unconfirmed to the user (see lines 18-19 in Figure 14.4).

Table 14.1 Beliefs in genuineness of violation observations of *Rule 1*

Event (e)	Bel(Genuine(e,[15,22],∅))	Bel(¬Genuine(e,[15,22],∅))	Confirmed
P1	$2\alpha_1 - \alpha_1^2$	0	YES
P2	0	$2\alpha_1 - \alpha_1^2$	NO

In the case of the example regarding the violation of *Rule 1*, the algorithm will report *P1: Happens(signal(Lap33),15,R(15,15))* as a confirmed predicate and *P2: ¬Happens(signal(Lap33),t,R(15,17))* as an unconfirmed predicate. This will be due to the beliefs in the genuineness and non genuineness of the events unified with these predicates which are shown in Table 14.1. It should be noted that in order to calculate the belief and disbelief in the genuineness of *P2*, the algorithm calculates the belief and disbelief in the genuineness of ¬*P2* assuming that there is an event of signal sent from the device *Lap33* at some time point from t=15 to t=17 in the event log.

14.6 Detection of S&D Threats

14.6.1 Overview

In some cases, the detection of violations of S&D monitoring rules after they occur might not be sufficient for restoring the system back into a secure and dependable state. This is because the required action for doing so can be expensive to take or because no such action may be possible. Thus, in addition to the detection of occurred violations, it is also important to be able to predict whether violations of S&D monitoring rules are likely to occur in some future state during the operation of a system. EVEREST provides support for predicting potential violations of monitoring rules, referred to as *threats*.

[5] Bel(P) and Bel(¬P) represent the proposition Bel(Genuine(P, $[T_{min},T_{max}]$, E_o)) and Bel(¬Genuine(P, $[T_{min},T_{max}]$, E_o)) respectively.

A threat in EVEREST is defined as a potential violation of an S&D monitoring rule and is associated with a belief measure indicating how likely the violation is, given the current state of the system that is being monitored. As an example of threat detection, consider the following rule for the LBACS system:

```
Rule 6:

∀ U:User; C1, C3: Device; C2: ControlServer; t1, t2:Time
Happens(e(_e1,_C1,_C1,REQ,login(_U,_C1),_C1),t1,ℜ(t1,t1)) ∧
Happens(e(_e2,_C2,_C2,REQ,login(_U,_C2),_C2),t2,ℜ(t1,t2)) ∧ _C1 ≠_C2
⇒ (∃ t3:Time)
Happens(e(_e3,_C1,_C1,REQ,logout(_U,_C1),_C1),t3,ℜ(t1+1,t2+1))
```

This rule states that if a user (*_U*) logs on some device (*_C1*) and later (s)he logs on to another device (*_C2*), by the time of the second login (t2), he/she must have logged out from the first device. Effectively, the rule prevents users from logging on to different devices simultaneously in order to reduce the scope for masquerading attacks. Allowing for simultaneous logging provides scope for such attacks since when a user *_U* is logged on to different devices simultaneously, it is possible to leave one of them unattended and then some other user may start using the device with *_U*'s credentials. Monitoring *Rule 6* and blocking logging attempts that violate it would prevent such cases. Also monitoring *Rule 6* would detect cases where some user who has got hold of the credentials of user *_U* tries to use them to log on with the identity of *_U* at the same when *_U* is logged on with his/her credentials from a different device.

Beliefs in the potential of threats with respect to *Rule 6* would need to be estimated in the following states of the monitoring process:

(a) When an event that matches *_e1* but no events that match *_e2* have been received by the monitor

(b) When an event that matches *_e2* but no event matching *_e1* has been received by the monitor

(c) When an event matching *_e1* and an event matching *_e2* have been received by the monitor

(d) When an event matching *_e1* and an event matching *_e2* have been received by the monitor and an event E has been received from the event captor that should have sent *_e3* at some time point t' > t2 indicating that *_e3* will not arrive. The absence of the *_e3* event could be derived from E in this case using the principle of *negation as failure* (NAF) − since from t' > t2 the monitor knows that it cannot receive any event with a timestamp earlier than t' and therefore earlier than t2.

In case (a) above, the threat likelihood for *Rule 6* will be a measure of the belief that the event *_e1* which has been matched with the rule is genuine, an event *_e2* matching the rule will occur within the time range (t1, t2], and no event matching *_e3* will occur in the range (t1, t2]. In case (b), the threat likelihood for *Rule 6* will be a measure of the belief that the event *_e2* which has already matched the rule is genuine and an event of type *_e1* matching the rule has already occurred within the time range *(latestTime(captor(_e1)), t2]* but not received by the monitor and an event matching *_e3* will occur in the range *(latestTime(captor(_e1)), t2]*. In case (c), the threat likelihood of *Rule 6* threat will be a measure of the belief that the *_e1* and *_e2* events that match the rule are genuine and an event of type *_e3* matching the rule will not occur in the time range (t1, t2]. Finally, in case (d), the threat likelihood of *Rule 6* will be a measure of the belief that the *_e1* and *_e2* events that match the rule are genuine and the event of type E which provided the basis for deriving ¬ *_e3* is genuine and, therefore, the application of the NAF principle in deriving ¬ *_e3* is valid.

The functions that we use to measure these beliefs and the ways of combining them to provide an estimate of the threat likelihood of the rule are discussed in the following.

14.6.2 Belief Functions

The calculation of threat likelihood requires the measurement and combination of beliefs of three different types: (i) basic probabilities in the genuineness of events that have been recorded in the log of the monitor (i.e., basic probabilities in the genuineness of events like *_e1* and *_e2* in case (c) above), (ii) basic probabilities in the occurrence of an event of a specific type within a time range that is determined by another event (i.e., basic probabilities of seeing an event like *_e2* after an event *_e1* has occurred as in case (a) above), (iii) basic probabilities in the validity of the derivation of the negation of an event when another event's occurrence indicates that the time range within which the former event should have occurred has elapsed (i.e., basic probabilities in events like ¬*_e3* given another event *E* as in case (d) above).

The calculation of basic probabilities in the genuineness of events is based on the function m^{EX} that we defined in Section 14.4.2 (see Definition 4). The second type of basic probability functions that we use in threat detection provides likelihood measures in the occurrence or not of an event E_i when another event E_j that E_i is temporally constrained by has occurred. This type of basic probability functions is defined as follows:

Definition 6: The basic probability in the occurrence of an event E_i within the time range determined by another valid event E_j, $m_{i|j}$, is defined as:

$$m_{i|j}(X) = \begin{cases} k_{ij} = \dfrac{\sum_{e_j \in Log(E_j)} m(e_j)[\sum_{I \in \wp(Log(E_i|e_j)) \wedge I \neq \phi} (-1)^{|I|+1} \prod_{e_k \in I} m(e_k)]}{\sum_{e_j \in Log(E_j)} m(e_j)} \\ if\ X = \{<e_1, \ldots, e_n> | e_j = 1, e_i = 1\} \\ k'_{ij} = \dfrac{\sum_{e_j \in Log(E_j)} m(e_j)[\sum_{e_i \in Log(E_i|e_j)} m(\neg e_i)]}{\sum_{e_j \in Log(E_j)} m(e_j)} \\ if\ X = \{<e_1, \ldots, e_n> | e_j = 1, e_i = 0\} \\ 1 - k_{ij} - k'_{ij} & if\ X = \theta \\ 0 & Otherwise \end{cases}$$

where

- *$Log(E_j)$* is a randomly selected sample of N events of type E_j in the event log up to the time point when $m_{j|i}$ is calculated
- *$Log(E_i|e)$* is the set of the events of type E_i in the event log that have occurred within the time period determined by *e* and up to the time point when $m_{i|j}$ is calculated
- $I \in \wp(Log(E_j|e))$ denotes any set in the powerset of $Log(E_j|e)$
- *m(e)* is the basic probability $m^{EX}(e)$ defined by Definition 4 in the case of non negated events E_j or the basic probability assignment $m^{NAF}{}_{j|u}$ defined in Definition 7 below for negated events $\neg E_j$

According to the above definition, $m_{i|j}(X)$ measures the basic probability of the occurrence of a genuine event of type E_i within the time range determined by events of type E_j, as the average belief of seeing a genuine event of type E_i within the time range determined by a genuine event of type E_j. More specifically, for each occurrence of an E_j event, $m_{i|j}(X)$ calculates the basic probability of seeing at least one genuine event of type E_i within the period determined by E_j. Assuming that the set of such E_i events is *$Log(E_i|e)$*, this basic probability is calculated by the formula $\Sigma_{I \subseteq Log(E_i|e) \text{ and } I \neq \varnothing} (-1)^{|I|+1} \{\Pi_{Ei \in I} m_i(E_i))\}$. The latter formula measures the basic probability of at least one of the events in *$Log(E_i|e)$* being a genuine event, i.e., an event that has at least one explanation confirmed by other events in the log of the system, and uses the basic probabilities of individual events $m_i(E_i)$ defined in Definition 4 above for positive events, or the basic probability $m^{NAF}{}_{j|u}$ for negative events $\neg E_j$ that are themselves established by a third event E_u (see Definition 7 below). Thus, $m_{i|j}(X)$ discounts occurrences of events of type E_i which are not considered to be genuine, and the higher the number of genuine events of type E_i within the period determined by an E_j event, the larger the basic probability of the occurrence of at least one genuine event of type E_i that it generates. It should also be noted that $m_{i|j}(X)$ takes into account the basic probability of the genuineness of each occurrence of an event of type E_j within the relevant period (i.e., $m_j(e)$) and uses it to discount the evidence arising from E_j events which are not assessed to be genuine.

The $m_{i|j}(X)$ basic probability function is similar to conditional probabilities in classic probability theory, in the sense that it provides a measure of the joint probability of seeing a genuine E_j event and at least one genuine E_i event in a given time period around E_j. As we have, however, discussed in Section 14.4.1, in EVEREST we do not use classic probabilities but basic probability assignments and beliefs as defined the DS theory. This is because in some cases during the monitoring of a system it is not possible to know with certainty whether an event has occurred or not. As discussed in Section 14.4.1, such cases arise due to communication channel delays – an event *E* may have occurred but not received by EVEREST yet when its occurrence needs to be established due to delays in the communication channel transmitting the event from its event captor to the framework. The D-S theory of evidence [18] enables the explicit treatment of such uncertain cases by allowing the award of a basic probability measure to the disjunction of the propositions denoting the occurrence and non occurrence of the event in question rather than to any of these propositions.

The basic probability functions that we have introduced above do not cover cases where the absence of an event is deduced by the NAF principle. As we discussed earlier, EVEREST uses this principle to deduce the absence of an event E (i.e. $\neg$ E) that is expected to occur within a specific time range [t_L, …, t_U] when it receives another event E' from the same event captor that should sent E with a timestamp t' that is greater than t_U (t' > t_U) and has not received E up to that point. Considering, however, that the event E' that would trigger the application of the NAF principle in such cases might not be a genuine event itself, it is necessary to estimate the basic probability of $\neg$E. This function that measures this basic probability is defined below:

Definition 7: The basic probability in the absence of an event E_i or, equivalently, $\neg E_i$ due to the application of the NAF principle when another event E_j occurs is defined as:

$$m_{j|i}^{NAF}(X) = \begin{cases} m_j(E_j) & \mathit{if}\ X = \{< e_1, ..., e_n > | e_j = 1, e_i = 0\} \\ 1 - m_j(E_j) & \mathit{if}\ X = \theta \\ 0 & \mathit{Otherwise} \end{cases}$$

where *$m_j(E_j)$* is the is the basic probability in the genuineness of the event E_j that is computed as defined in Definition 4.

14.6.3 Combination of Beliefs through Belief Networks

The computation of the threat likelihood of a rule at runtime requires the combination of basic probability functions of the types introduced in Section 14.6.2. The exact combinations that will be required in each stage are determined by the following basic principles:

(i) The principle that the computation of the threat likelihood of a rule should be based on all the events that have been received by the monitor and can be matched with the rule or used to derive the absence of a negated event in it.

(ii) The principle that the computation of the basic probability of the occurrence or not of an event that has not been received yet by the monitor should be based on the combination of the smallest possible number of basic probability functions conditioned by events which the event in question has a direct temporal dependency upon in the rule (an event e_i is said to be directly temporally depend upon another event e_j if the upper or lower bound of the values of the time variable of e_i is defined by a formula that contains the time variable of e_j).

(iii) The principle that if alternative estimates of the basic probability of an event and its absence can be derived by different combinations of basic probability functions, the maximum of these estimates should be taken into account.

Based on the above principles, the different combinations of basic probability functions that would be required in the case of *Rule 6* depending on the set of events that have been received by the monitor at different stages of the monitoring process are summarised in Table 2. As shown in the table, for example, when the monitor has received an event matching *_e1* but not an event matching *_e2* and the absence of an *_e3* event can be established by the NAF principle (i.e., case 3 in the table), the threat likelihood of the rule would be estimated by the combination of the basic probability functions $(m_1 \oplus m_{2|1}) \oplus m^{NAF}_{3|u}$. Also, in cases where an event matching *_e1* and an event matching *_e2* but no event enabling the derivation of the absence of *_e3* by the NAF principle has occurred (i.e., case 2 in the table), the threat likelihood of *Rule 6* would be estimated by taking the maximum estimate generated from the combinations of the basic probability functions $(m_1 \oplus m_2) \oplus m_{3|1}$ and the basic probability functions $(m_1 \oplus m_2) \oplus m_{3|2}$.

Table 14.2 Derivation of threat likelihood in the case of *Rule 6*

#	Event *_e1*	Event *_e2*	Event ¬*_e3*	Required Combinations
1	received	received	derived by NAF	$(m_1 \oplus m_2) \oplus m^{NAF}_{3\|u}$
2	received	received	not derived by NAF	$(m_1 \oplus m_2) \oplus m_{3\|2}$ $(m_1 \oplus m_2) \oplus m_{3\|1}$
3	received	not received	derived by NAF	$(m_1 \oplus m_{2\|1}) \oplus m^{NAF}_{3\|u}$
4	received	not received	not derived by NAF	$(m_1 \oplus m_{2\|1}) \oplus m_{3\|1}$
5	not received	received	derived by NAF	$(m_2 \oplus m_{1\|2}) \oplus m^{NAF}_{3\|u}$
6	not received	received	not derived by NAF	$(m_2 \oplus m_{1\|2}) \oplus m_{3\|2}$
7	not received	not received	derived by NAF	Cannot be estimated
8	not received	not received	not derived by NAF	Cannot be estimated

It should be noted that the operator ⊕ in Table 2 denotes the combination of two basic probability assignments by applying the rule of the *orthogonal sum* of the DS theory. According to this rule, a basic probability function m_1 can be combined with another basic probability function m_2 if m_1 and m_2 assign basic probability measures to at least two overlapping sets of propositions (aka *focals*) and the combined basic probability that is generated by the combination of *m1* and *m2* is given by the formula $m_1 \oplus m_2 (P) = (\Sigma_{X \cap Y = P}\, m_1(X) \times m_2(Y)) / (1 - k_0)$ where k_0 is a normalising parameter used to increase the basic probability assigned to the non-empty intersections of the focals of m_1 and m_2 in proportion to the basic probability that would be assigned to the empty intersections of these focals that is computed as $k_0 = \Sigma_{V \cap W = \varnothing \text{ and } V \subseteq \theta \text{ and } W \subseteq \theta}\, m_1(V) \times m_2(W)$.

To represent the different cases of combining basic probability functions in order to calculate the threat likelihood of a monitoring rule, EVEREST constructs a graph having vertices representing the different events in the rule and *directed, labelled* edges between them indicating dependencies between the time variables of these events. The edges in this graph are derived from the time variables which constrain the occurrence of each event, and indicate how evidence can be propagated at runtime by combining the different basic probability assignments that are associated with the observed events. This graph is called “belief graph” and is constructed by the algorithm of Figure 14.5.

```
Construct_DS_Belief_Graph(R, DSG_R)
1.  find all n events $e_i$ in R /* R is assumed to be the negated form of a monitoring rule or an attack signature */
2.  construct a node representing the starting point in the assessment of the threat likelihood of R, called "Start" node.
3.  for each event $e_i$ ($i \le n$) do:
4.          construct a node for $e_i$ and store the mapping of the time variable of this event as $M(t_i)= e_i$ (i.e. store the fact that time variable $t_i$ has been used to declare the time of occurrence of event $e_i$.)
5.          build a list $TVARS_i$ of all time variables $t_k$ appearing in the lower and upper bound of the time variable $t_i$ of $e_i$.
6.  end for
7.  for each event $e_i$ ($i \le n$) do:
8.          for each time variable $t \in TVARS_i$ such that $t \neq t_i$ do:
9.              Construct an edge to $e_i$ from $e_p=M(t)$, labelled by $m_{i|p}$, i.e., the basic probability of observing (or not) $e_i$ given $e_p$.
10.             if $e_i$ is not a negated event then
11.                 Construct an edge from $e_i$ to $e_p=M(t)$, labelled by $m_{p|i}$, i.e., the basic probability of observing (or not) $e_p$ given $e_i$.
12.             end if
13.         end for
14.         if $e_i$ is not a negated event then
15.                 construct an edge from the "Start" node to $e_i$, labelled by the basic probability $m_i$ of $e_i$.
16.         else /* negated events */
17.                 if $e_i$ has a time range defined by constant values then
18.                     construct an edge from the "Start" node to $e_i$, labelled by the basic probability $m^{NAF}_{i|<x>}$ of $e_i$
19.                 end if
20.         end if
21. end for
end Construct_DS_Belief_Graph
```

Fig. 14.5 Algorithm for constructing D-S belief graphs

This algorithm initially identifies the events of a given monitoring rule R and constructs a *start* node to represent the starting point of the accumulation of evidence at runtime (see line 2). Then for each event in the rule, it constructs a node to represent the occurrence of the event at runtime (line 4) and identifies the dependencies of the event to other events (line 5). At this step an event E_j is taken to depend on all other events E_i whose time variables appear in the expressions that define the lower and upper bound of the time variable of E_j. After identifying these dependencies, the algorithm creates a directed edge from all the events E_i that E_j depends on to E_j (see line 9). These edges indicate the paths for obtaining a basic probability for E_j when any of the events E_i is observed. Also an opposite edge from E_j to each of the events E_i is created provided that E_j is not a negated event (see lines 10-12). The latter edges will be used when E_j is observed before E_i in order to indicate how the basic probability of E_i can be computed given E_j.

Note also that no backward edges are constructed from an event E_j to the events that it depends on, if E_j is a negated event (see condition in line 10). This is because, in EVEREST, negated events can only be derived through the application of the NAF principle when their ranges have fully determined boundaries (an event expected in a fully determined time range [a,b] is known to not have happened when the monitor receives the first event from its captor with a time stamp *t*

> *b* without having received the event itself up to that point). Fully determined boundaries, however, will not be possible to have for E_j unless E_i has already occurred. Hence, it will not be possible to derive the truth value of E_j before that of E_i and therefore compute a basic probability assignment for the latter event based on the basic probability of the former. The label attached by the algorithm on an edge from an event E_i to an event E_j will be $m_{i|j}$, i.e., it will represent the basic probability assignment of observing (or not) E_j given that E_i has already been observed (note that this is different to $m_{j|i}$).

Following the generation of edges between events, the algorithm constructs edges to link the *Start* node of the graph with the nodes representing the different non negated events of the rule (see lines 14–20). These edges are labelled by the basic probability assignment corresponding to the event E_i that they point to (i.e., the basic probability assignment m_i). Negated events, on the other hand, are linked with the *Start* node only if they have a time range defined by constant values at the time of application of the algorithm (i.e., prior to runtime) and, therefore, it will be possible to establish their absence or not prior the seeing any other event at runtime (see conditions in lines 14 and 17)[6]. The edge linking the *Start* node with a negated event E_i is labelled by the basic probability function $m^{NAF}_{i|<x>}$. This function is partially determined as it includes the placeholder <x>. At runtime this placeholder will be bound to the identifier of the event E_j that triggers the application of the NAF principle to derive the absence of E_i creating a fully determined basic probability function $m^{NAF}_{i|j}$ which will be used to estimate the basic probability of $\neg E_i$.

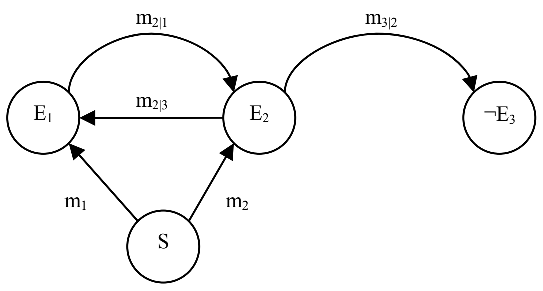


Fig. 14.6 D-S belief graph for *Rule 6*

An example of a DS belief graph is shown in Figure 14.6. This graph has been constructed to express the dependencies and the different paths of combining beliefs for *Rule 6*. The graph reflects that the occurrence of E_2 in the rule depends on the occurrence of E_1 since the range of the time variable of E_2 (i.e., $\Re(t_1,t_2)$) refers to the time variable of E_1 but not vice versa (the range $\Re(t_1,t_1)$ of t_1 indicates that E_1 is an event with a not constrained time variable). Thus, an edge from E_1 to E_2

[6] Such events may typically appear in rules of the form ¬Happens(e1,t1,R(a,b)) ⇒ Happens(e2,t2,R(t1,t1+c)). The event ¬Happens(e1,t1,R(a,b)) in this rule has a time range with fully determined boundaries (a and b) prior to runtime and will remain as a negated event in the negated form of the rule, i.e., ¬Happens(e1,t1,R(a,b)) ∧ ¬Happens(e2,t2,R(t1,t1+c))

labelled by $m_{2|1}$ has been inserted in the graph as well as another edge from E_2 to E_1 labelled by $m_{1|2}$. Similarly, as the time range of the event $\neg E_3$ (i.e., $\Re(t_1+1,t_2-1)$) refers to the time variables t_1 and t_2 of the events E_1 and E_2, the graph contains edges from E_1 to $\neg E_3$ and E_2 to $\neg E_3$. Note, however, that the graph does not contain an edge from $\neg E_3$ to E_2 or from $\neg E_3$ to E_1 as the former event cannot be derived by NAF unless E_1 and E_2 are received first. Finally, the graph includes edges from the starting node to E_1 and E_2. These edges are labelled by m_1 and m_2 representing the basic probability functions that are to be used when the occurrence or absence of the events E_1 or E_2 is established from the starting node.

```
Compute_Threat_Likelihood(Ei, DSGR, R)
 1.  find the sets of the known events KE and the set of the unknown events UE in DSGR
 2.  m = basic_probability_assignment(<start, Ei>)
 3.  CombinedBPA = {}
 4.  for each Ek in KE do /* combine the BPAs of events in KE */
 5.      m = m ⊕ basic_probability_assignment(<start, Ek>)
 6.      CombinedBPA = CombinedBPA ∪ basic_probability_assignment(<start, Ek>)
 7.  end for
 8.  for each ej ∈ UE do
 9.      insert all the paths from ei to ej, which do not include any event in KE, into Pij
10.      for each p ∈ Pij do
11.          for each edge L in p do /* combine the BPAs of paths to unknown events */
12.              if basic_probability_assignment(L) ∉ CombinedBPA then
13.                  m = m ⊕ basic_probability_assignment(L)
14.                  CombinedBPA = CombinedBPA ∪ basic_probability_assignment(L)
15.              end if
16.          end for
17.      end for
18.  end for
19.  mark Ei as a known event in DSGR
20.  return (m(events(¬R), m(events(R)))
end Compute_Threat_Likelihood
```

Fig. 14.7. Algorithm for computing threat likelihood

At runtime, belief graphs are used to record the events matched with a given monitoring rule and determine the combination(s) of basic probability functions that will be needed to compute the threat likelihood of the rule. In general, given a set of received and a set of unknown (i.e., not received/derived) events, the threat likelihood of a rule is evaluated by combining the basic probabilities of the received events that match with rule (these basic probabilities label transitions from the *Start* node of the graph to the nodes that match with the received events) and the conditional beliefs on the unknown events. It should be noted that in such cases, there may be more than one known events in the graph which are linked directly with the unknown one. In such cases, the conditional beliefs of the unknown events $m_{j|i}$ are computed by considering all paths which start from some known event e_i and end in the unknown event e_j, without passing through any other known events (this ensures that known events will not be considered as supporting evidence for unknown ones multiple times). The algorithm for evaluating the threat given a belief network is shown in Figure 14.7.

14.6.4 Example of Threat Detection

To demonstrate the estimation of the threat likelihood of a rule in EVEREST consider *Rule 6* again and the following sequence of events:

- *Happens(e(e100,Lap30,Lap30,REQ, login(User1,Lap30),Lap20),80,ℜ(80,80))*
- *Happens(e(e101,Lap2,Lap2,REQ, login(User1,Lap2),Lap2),87,ℜ(87,87))*

When it arrives at EVEREST, the first of these events (*e100)* can be matched with the nodes E1 or E2 of the belief graph of Figure 14.6. Each of these matches will produce a separate instantiation of the belief graph and lead to the estimation of different threat likelihoods. When matching *e100* with node E1 in the graph of Figure 14.6, the threat likelihood of *Rule 6* will be computed by combining the basic probability functions: $(m_1 \oplus m_{2|1}) \oplus m_{3|1}$.

Based on the generic definitions of these functions, in Section 14.6.2, it can be shown that the application of the rule of the orthogonal sum will result in the following functional form for $(m_1 \oplus m_{2|1}) \oplus m_{3|2}$:

$$(m_1 \oplus m_{2|1} \oplus m_{3|1}(E_1 \wedge E_2 \wedge \neg E_3)) = \frac{k'_{31}k_{21}k_1 \ + k'_{31}k_1\ (1-k_{21}-k'_{21}) + k'_{31}k_{21}(1-k_1 \ -k'_1)}{1-(k'_{31}k'_{21}(1-k'_1)+k'_{31}k'_{21}(1-k'_1)}$$

Thus, assuming that

- the basic probability assignments to the genuineness and non genuineness of *e100* are $k_1 = 0.8$ and $k_1' = 0.1$, respectively;
- the conditional basic probability assignments in observing and not observing a second genuine login event within L=100 time units after the observation of *e100* are $k_{21} = 0.6$ and $k_{21}' = 0.4$, respectively; and
- the conditional basic probabibility assignment of not observing a genuine logout event in the period of L=100 time units between two genuine login events are $k_{31} = 0.2$ and $k_{31}'=0.6$, respectively

the threat likelihood for the first instance of the rule will be:

$$(m_1 \oplus m_{2|1} \oplus m_{3|1}(E_1 \wedge E_2 \wedge \neg E_3)) = \frac{0.6*0.6*0.8+0.6*0.8*0+0.6*0.6*1}{1*(0.2*0.4*0.9+0.6*0.4*0.9)} = 0.45$$

The threat likelihood estimate for the same rule instance will be updated when the event *e101* arrives at EVEREST. Upon its arrival, *e101* will be matched with the node E2 in the above belief graph instance. Thus, according to the *Compute_Threat_Likelihood* algorithm of Figure 14.7, the threat likelihood will be estimated by the combination of the basic probability assignments $(m_1 \oplus m_2) \oplus m_{3|2}$. The application of the rule of the orthogonal sum of the DS theory for this combination will result in the following formula:

$$(m_1 \oplus m_2) \oplus m_{3|2}(E_1 \wedge E_2 \wedge \neg E_3)) = \frac{k'_{31}k_2k_1 \ + k'_{31}k_1\ (1-k_2-k'_2) + k'_{31}k_2(1-k_1 \ -k'_1)}{1-(k'_{31}k'_2(1-k'_1)+k'_{31}k'_2(1-k'_1)}$$

Thus, if the basic probability assignments in the genuineness of *e101* (i.e., *m_2(Genuine(e101,…)*) and the non genuineness of this event (i.e., *m_2(¬Genuine(e101,…)*) are k_2 =0.8 and k_2' = 0.2 respectively, and the threat likelihood will be:

$$(m_1 \oplus m_2) \oplus m_{3|1}(E_1 \wedge E_2 \wedge \neg E_3)) = 0.504$$

The increase in the threat likelihood in this case is due to the fact that the basic probability of E2 given by m_2(X) is higher than the basic probability of E2 that is computed by the combination $m_1 \oplus m_{2|1}$ (0.8 vs. 0.53).

14.7 Related Work

In the context of software system monitoring, diagnosis focuses on the detection of the reasons for system failures. Diagnosis typically involves the identification of trajectories of system events that have led to a failure (i.e., problematic events) using automata whose purpose is to recognize faulty behaviour [6,13,17]. More specifically, in [6] diagnosis is carried through the synchronization of automata modelling the expected behaviour of the system that is being monitored and the events captured from it. In [13] a similar but incremental approach is adopted. In this approach, synchronization is initially performed for individual system components and the partial synchronised models are aggregated for the global system.

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 [21]) but the explanation of such causes in the presence of incomplete and/or not trusted event traces. Our approach draws upon work on temporal abductive reasoning [3, 15, 20] and its applications to diagnosis [2, 14]. However, it is based on a newly developed algorithm for abductive search using Event Calculus. This algorithm generates all the possible explanations of a formula (unlike [3, 20]) and then computes beliefs in the validity of individual explanations using the DS theory [18].

The approach that has been undertaken for threat detection as part of the SERENITY monitoring framework is related to earlier work on *intrusion detection* [5, 12]. It should be noted, however, that most of the existing intrusion detection systems can detect only malicious actions that have already happened (aka intrusions). Our approach is different from this work since it views threat detection as the problem of trying to forecast actions that would violate certain S&D properties expressed as monitoring rules.

Existing approaches to intrusion detection have been distinguished into *anomaly-based* and *misuse-based* [12]. Anomaly-based approaches [1, 5, 10] assume that attacks involve some abnormal behaviour of the system that is being monitored. Intrusions are, thus, detected as deviations from the expected normal behaviour of the system. Misuse-based approaches [7, 11, 27], on the other hand, are based on models of known attacks.

The threat detection approach that has been undertaken in SERENITY can be seen as an anomaly-based approach. Furthermore, since threats are detected as deviations from a model of normal behaviour (that is expressed by the rules which are specified in S&D patterns and are being monitored), our approach can also be classified as *model* or *specification*-based [1, 10]. Finally, we should note that our approach has some similarity with statistical approaches to intrusion detection based on Bayesian networks (e.g. [27]). The difference from these approaches is that we use Dempster Shafer beliefs to provide measures of the genuineness of individual events and the likelihood of potential rule violations due to some inherent uncertainty about the occurrence or not of specific events which arises by communication delays between event sources and the reasoning system that performs the threat analysis.

14.8 Conclusions

In this chapter we have presented the mechanisms that are incorporated in the monitoring framework of SERENITY to support the provision of diagnostic information for violations of S&D monitoring rules and the detection of potential violations of such rules.

The diagnosis of violations of S&D rules is based on the generation of possible explanations for the runtime events which are involved in the violations of the rules through the use of *abductive reasoning*. This process deploys the *assumptions* that have been specified about the system that is being monitored and the S&D Patterns which are deployed by it. The possible explanations which are generated by this process are checked against other runtime events to establish if there is further evidence about the validity of the explanations. The input to this search process includes the expected *consequences* of the abduced explanations and preconditions that need to be satisfied for an explanation to be able to entail the event in question.

The EVEREST diagnosis framework computes beliefs in the validity of the explanations of the events that are involved in S&D violations, based on matches that may be found between the preconditions and consequences of these explanations with other runtime events and a recursive assessment of the genuineness of these events. Ongoing work on diagnosis focuses on conducting an experimental evaluation of the undertaken approach.

The detection of potential violations of S&D properties in the SERENITY monitoring framework is also based upon the computation of beliefs that violations of such properties are likely to occur. The computation of such beliefs is based upon the diagnostic mechanisms of EVEREST which provide the basic assessment of the genuineness of the events received by the framework and historical data about the frequency of co-occurrence of events which are connected by temporal constraints within specific S&D monitoring rules. These historical data provide the basis for computing beliefs in the potential occurrence or not of an event when another event that it is constrained by has occurred and is known to be genuine.

References

1. Chari SN and Cheng PC (2003) Bluebox: a policy-driven, host-based intrusion detection system. ACM Trans. Inf. Syst. Security 6(2): 173-200
2. Console L, Terenziani P, Dupré DT (2002) Local Reasoning and Knowledge Compilation for Efficient Temporal Abduction. IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2002.1047764
3. De Kleer J, Williams BC (1987) Diagnosing Multiple Faults. Artificial Intelligence 32(1), 97-130
4. Denecker M, De Schreye, D (1992) Temporal Reasoning with Abductive Event Calculus. In: Neumann B (ed) Proc. of the 10th European Conference on Artificial Intelligence, 384-388
5. Denning D (1987) An Intrusion Detection Model. IEEE Transactions on Software Engineering, 13(2): 222-232.
6. Grastien A, Cordier M, Largouët C (2005) Incremental Diagnosis of Discrete-Event Systems. In Proc. of 15th Int. Work. On Principles of Diagnosis (DX05)
7. Ilgun K, Kemmerer RA, and Porras PA, (1995) State Transition Analysis: A Rule-based Intrusion Detection System. IEEE Transactions on Software Engineering, 21(3): 191-199.
8. Kloukinas C, Ballas C, Presenza D, Spanoudakis G (2006) Basic set of Information Collection Mechanisms for Run-Time S&D Monitoring. Deliverable A4.D2.2, SERENITY Project, http://www.serenity-forum.org/IMG/pdf/A4.D2.2_informationCollectionMechanism_v0.15_final_e.pdf Accessed 29 November 2008
9. Knight K (1989) Unification: a multidisciplinary survey ACM Computing Surveys, 21(1):93-124 http://www.isi.edu/natural-language/people/unification-knight.pdf Accessed 29 November 2008
10. Ko C, Ruschitzka M, and Levitt K (1997) Execution monitoring of security-critical programs in distributed systems: a Specification-based approach. In IEEE Symposium on Security and Privacy (SP '97), 175-187.
11. Kumar S and Spafford EH (1994) A Pattern Matching Model for Misuse Intrusion Detection. In Proc. of 17th National Computer Security Conference. 11-21.
12. Lazarevic A, Kumar V, Srivastava J (2005) Intrusion detection: a survey In: Managing cyber-threats: issues approaches & challenges, Available from Springer
13. Mahbub K, Spanoudakis G, Kloukinas C (2007) V2 of dynamic validation prototype. Deliverable A4.D3.3, SERENITY Project. Available from: http://www.serenity forum.org/IMG/pdf/A4.D3.3_-_V2_of_Dynamic_validation_Prototype.pdf Accessed 29 November 2008
14. Pencolé Y, Cordier M (2005) A formal framework for the decentralised diagnosis of large scale discrete event systems & its application to telecommunication networks, Artificial Intelligence 164: 121-180
15. Poole D (1989) Explanation and prediction: an architecture for default and abductive reasoning, Computational Intelligence 5(2): 97-110
16. Ray O, Kakas A (2006) ProLogICA: A Practical System for Abductive Logic Programming.11th Int. Workshop on Non-monotonic Reasoning, 304-312
17. Reiter R (1987) A theory of diagnosis from first principles, Artificial Intelligence, 32(1): 57-96
18. Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis DC (1996) Failure diagnosis using discrete-event models. IEEE Transactions on Control Systems Technology, 4(2):105-124
19. Shafer G (1975) A Mathematical Theory of Evidence. Princeton University Press
20. Shanahan M (1999) The Event Calculus Explained. In Artificial Intelligence Today, LNAI 1600:409-430
21. Shanahan M (2000) Abductive Event Calculus Planner, Journal of Logic Programming 44: 207-239
22. Spanoudakis G, Mahbub K (2006) Non intrusive monitoring of service based systems. Int. J. of Cooperative Information Systems, 15(3):325–358

23. Spanoudakis G, Tsigkritis T (2008) 1st Version of Diagnosis Prototype. Deliverable A4.D5.1, SERENITY Project, Available from: http://www.serenity-forum.org/IMG/pdf/A4.D5.1_first_version_of_diagnosis_prototype_v1.1_final.pdf Accessed 29 November 2008
24. Spanoudakis G, Tsigkritis T (2008) 2nd Version of Diagnosis Prototype. Deliverable A4.D5.2, SERENITY Project, Available from: http://www.serenity-forum.org/IMG/pdf/A4.D5.2_first_version_of_diagnosis_prototype_v1.1_final.pdf Accessed 29 November 2008
25. Tsigkritis T, Spanoudakis G (2008) Diagnosing Runtime Violations of Security and Dependability Properties. In Proc. of 20th Int. Conference in Software Engineering and Knowledge Engineering, 661-666
26. Tsigkritis T, Spanoudakis G (2008) A temporal abductive diagnosis process for runtime properties violations, ECAI 2008 Workshop on Explanation Aware Computing
27. Valdes A and Skinner K (2000) Adaptive, Model-based Monitoring for Cyber Attack Detection. In Recent Advances in Intrusion Detection (RAID 2000), LNCS 80-92. Springer.