Ceaseless CBR

Francisco J. Martin[1] and Enric Plaza[2]

[1] School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, 97331 OR, USA
fmartin@cs.orst.edu
[2] IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain
enric@iiia.csic.es

Abstract. Most CBR systems try to solve problems in one shot neglecting the sequential behavior of most real world domains and the simultaneous occurrence of interleaved problems proper to multi-agent settings. This article provides a first answer to the following question: how can the CBR paradigm be enriched to support the analysis of unsegmented sequences of observational data stemming from multiple coincidental sources? We propose Ceaseless CBR, a new model that considers the CBR task as on-going rather than one-shot and aims at finding the best explanation of an unsegmented sequence of alerts with the purpose of pinpointing whether undesired situations have occurred or not and, if so, indicating the multiple responsible sources or at least which ones are the most plausible.

1 Introduction

In an ever-increasing diversity of domains such as intrusion detection, forecasting conflicts in international event analysis, fraud detection in cellular telephones, etc, automated sensors (in addition to being noisy and imperfect) lack the intelligence to disambiguate (differentiate and synthesize) the parts corresponding to distinct problems so they resort to piecing all the sensed parts together into only one sequence. That is, several problem descriptions corresponding to problems occurring in parallel are serialized into a unique sequence without comprehensible criteria for its further understanding. Most CBR systems commonly presuppose individualized problem descriptions with well-specified boundaries that encompass all the information needed to solve the current problem in only "one shot". This assumption makes impracticable their direct deployment in the above domains. The fact of the matter is that most CBR systems, built on the dominant mainstream CBR model, are devised bearing in mind the following three interrelated assumptions: (i) *non-coincidental sources*: there is only a problem occurring at a time. Said differently, problems are solved successively one after another without considering problems that concur and whose origin

could be related and could require a joint solution; (ii) *full-fledged problem descriptions*: a problem description is provided in only one shot (instantaneous situations) with well-defined and clear limits—i.e., the boundaries of each case are perfectly delimited; and (iii) *individual cases independency*: cases are manipulated (retrieved, reused, revised or retained) in isolation without contemplating their sequential (spatial or temporal) structure or (serial or parallel) relationship with other cases in the past. That is, they assume (snapshot) cases that are independent of each other, and therefore relations among cases (e.g., sequential relationships) are not taken into account.

These assumptions make CBR unsuitable for a number of challenging problems—mainly those that involve temporally-evolving sequences of observational data. Thus our interest in investigating new techniques that allow one to alleviate such constraints and applying CBR in a variety of much more complex domains. In this article we give a first answer to the following question: how can the CBR paradigm be enriched to support the analysis of unsegmented sequences of observational data stemming from multiple coincidental sources? We propose Ceaseless CBR a new CBR model that aims at finding the best explanation of an unsegmented sequence of alerts with the purpose of pinpointing whether undesired situations (an attack, fault, etc) have occurred or not and, if so, indicating the multiple responsible sources (if more than one intervened) or at least which ones are the most plausible. Moreover, Ceaseless CBR prioritizes each individual alert according to the proposed explanations.

This article proceeds as follows. We initially describe the concrete application domain where we have evaluated Ceaseless CBR in Sec. 2. Sec. 3 puts the work covered in perspective. We see Ceaseless CBR as a constructive situation awareness process governed ceaselessly by *observational data*, *sequential cases*, and *case activations*. We discuss each of these concepts in detail in Sec. 4, Sec. 5, and Sec. 6 respectively. The description of Ceaseless Retrieve and Ceaseless Reuse in Sec. 7 and Sec. 8 constitute the bulk of this article. Finally, Sec. 9 concludes the article with a succinct discussion about Ceaseless CBR.

2 Application Domain

We have conducted an exploratory analysis of Ceaseless CBR in *intrusion detection*, concretely in *alert triage*—the rapid and approximate prioritization for subsequent action of an Intrusion Detection System (IDS) alert stream [1]. The fact of the matter is that current IDSes generate an unmanageable number of false positive alerts[3] which in turn increases the difficulties for the proper identification of real and malicious attacks. Security managers are so overwhelmed that they frequently disable the alert device due to the consistent assumption that nothing is wrong reinforced by the fact that the alert device "cried wolf" too often. There are those who even postulate that current IDSes not only have failed to provide an additional layer of security but have also added complexity

[3] Alerts signaled when there is a manifest absence of intrusive behavior.

to the security management task. Therefore, there is a compelling need for developing a new generation of tools that help to automate security management tasks such as alert triage. Generally speaking, two kinds of components can be distinguished in the current state of the art IDSes [2]: *probes* and and *aggregation and correlation components* (ACCs). Probes compile information using host-based sensors as well as network-based sensors and evoke an alert whenever suspicious activity is detected. Probes can be considered low-level sensors such as firewalls or integrity checkers. Snort is a representative example of a signature-based sensor that we have used for our experiments. Snort performs lightweight real-time traffic analysis and packet logging on IP networks [3]. An ACC takes as input alerts from probes and after analyzing and correlating received alerts decides whether to send such alerts to the network manager or not [2]. Ceaseless CBR aims at increasing the performance of such decisions. The different techniques devised throughout this work have been embodied within a research prototype—called Alba (Alert Barrage)—that could be cataloged as a striking example of these components [1].

We have constructed three data sets for measuring the performance of our techniques using *honeypots* to compile alerts in three different real-world scenarios. The Rustoord data set consists of 31483 alerts with an average frequency of 1968 alerts/week. The Naxpot data set contains 204977 alerts with an average frequency of 5856 alerts/week. The Huckleberry data set is composed of 219886 alerts with an average frequency of 13743 alerts/week. We have used the ROC[4] evaluation framework described elsewhere to analyze the performance of Ceaseless CBR in the above data sets [1]. Our results showed a significant increase in performance, as measured by ROC AUC (Area Under the Curve) [1]. Ceaseless CBR was able to keep the true positive rate over and above 99% and the false positive rate under and below 1%. We have achieved significant reductions in the weekly alert load. We got reductions up to a 95.90% in Rustoord data-set, to 80.89% in Naxpot, and to 93.02% in Huckleberry data-set. Our evaluations demonstrated how a Ceaseless CBR-enhaced IDS system is not only able to significantly reduce the weekly alert load but also to keep the number of false negatives very low and an admissible rate of false positives. This level of performance demonstrates that Ceaseless CBR can perform sufficiently for real world deployment.

3 Related Work

Four of the main and unusual CBR issues that we deal with in this work were partially opened, most of them early in the 90s, by separate seminal works [5–9]. Shavlik was the first to notice that most CBR systems usually presuppose *well-defined current situations*—situations where the boundaries of the current

[4] The term ROC (Receiver Operating Characteristic) refers to the performance (the operating characteristic) of a human or mechanical observer (the receiver) that has to discriminate between radio signals contaminated by noise (such as radar images) and noise alone [4].

case are cleanly defined [6]. Ram and Santamaría observed with much truth that CBR is mostly deployed as a high-level problem solving paradigm where situations are represented using discrete and static symbolic representations [8]. Ceaseless CBR is closely-related to Continuous CBR. Both methods need to provide a timely response to a time-varying situation (i.e., continuous on-line performance). While Continuous CBR practically operates in real-time Ceaseless CBR only aspires to work on a quasi-real time basis. This is due to the fact that we have to evaluate time-evolving sequences of complex objects as opposed to only vectors of analog values as Continuous CBR does. The input of Ceaseless CBR are unsegmented sequences of events dispersed over time and the task is to segment the sequence to provide the best explanation of the current situation and suggest an action. In Continuous CBR the current situation is given by a series of equally time-spaced real values and the task is to directly execute the actions. A drawback of Continuous CBR is that continuous cases are neither easily-interpretable by a human nor easy-to-integrate with higher-level reasoning and learning methods. Jacynski observed that most CBR approaches only cope with *instantaneous situations* (aka snapshot cases [10]) [9] and only few CBR systems deal with *time-extended situations* (aka time-dependent situations [10]). An instantaneous situation is a finite set of data that represents the state of the world at a particular point in time whereas a time-extend situation reflects the evolution of the world either through a continuum of instantaneous situations along a specific time line or through a sequence of events like Ceaseless CBR. Only a few additional CBR works have dealt with time-extended situations, the most noticeable being the work due to Jaere et al [10] who introduced *temporal cases* as a method for representing time-dependent situations within a knowledge-intensive CBR framework. Relatively little attention has been spent on CBR systems that are able to combine relevant pieces of several past cases when solving a new problem. The pioneering works in this aspect are Barletta et al [5] and Redmond [7].

The Ceaseless CBR inference process can be considered as an instantiation model of the inference process of parsimonious covering theory. In a nutshell, parsimonious covering theory is able to formalize many imprecise and intuitive aspects of abduction providing a good theoretical foundation for automated diagnostic problem-solving [11]. Traditional knowledge-based troubleshooting techniques such those used by rule-based systems or model-based systems cannot precisely capture the dynamic complexity of large systems, and thus CBR emerges as a suitable paradigm to do so [12, 13]. Lewis extended a Ticket Troubleshooting System (TTS) system with CBR methods that aided in computer network alarm management [12]. Gupta introduced SPOTLIGHT, a CBR tool for complex equipment troubleshooting [13]. Breese and Heckerman defined a decision-theorectic methodology for developing diagnosis and troubleshooting applications based on CBR [14]. They represented diagnostic cases by means of a specific belief network structure where nodes represented *issues*, *causes* and *symptoms*. This approach is similar in essence to Ceaseless CBR. However, in our approach a sequential case only stores part of the complete model for

problem determination that helps to determine its plausibility given a collection of alerts. Since we store sequential cases individually, we avoid on-the-fly construction for each new problem. Moreover, we consider a number of distinct problems (attacks) occurring coincidentally whereas they solved problems sequentially (one-by-one) supposing the occurrence of only one problem at a time. Their input is provided by an user and they used a myopic approximation (i.e., they presupposed that the user made just one observation at a time) whereas we receive the input from an automated process and deal with a sequence of interlaced observations corresponding to simultaneous problems. Conversational CBR also assumes partial rather than complete problem descriptions. However, when only a partial problem description is provided, an interactive dialogue is engaged with the user to better delimit and clarify the descriptions provided [15]. Through this conversation with the user, a complete and individual description is obtained in the end. Cunningham and Smith [16] proposed an incremental case retrieval technique based on a simple information theoretic metric to find the feature that best discriminates between the current set of retrieved cases and produce focused questions in electronic fault diagnosis.

To the best of our knowledge, only a few case-based approaches to intrusion detection have been published [17, 18]. Esmaili et al proposed a Case-Based Intrusion Detection System (CBIDS) whose input was the audit trail produced by an operating system and whose output was a collection of countermeasure actions that the system performed based on the severity of the intrusion detected so far [17]. Recently Schwartz et al proposed to improve the capabilities of Snort IDS [3] using a case-based approach [18]. They proposed a similarity measure based on a collection of distinct comparators for each feature (Snort rule properties) rather than using a complete match on all features as Snort. There are two main differences to our approach. First, we work at higher-level of abstraction using alerts provided by Snort as input and providing a priority as output whereas they used directly suspect network packets as input and provided alerts as output. Second, their approach was stateless since they assessed the danger of each suspect network packet (case) individually whereas our approach can be considered stateful and considers a whole sequence of alerts before determining the priority of individual alerts.

4 Observational Data

We suppose alerts that are triggered by automated real-time systems that collect and interpret sensor data in real-time. Alerts are complex objects made up of a set $\mathcal{F}$ of numeric, qualitative and structured features. We model alerts using *feature terms* that organize concepts into a hierarchy of *sorts*, and represent *terms* or *individuals* as collections of *features* (functional relations). For further details and examples see [1]. We assume that at a given point in time t there is a sequence of n alerts (alert stream) in the system. We denote by $\boldsymbol{S}^{(t)}$ the sequence of alerts received so far. Each alert ψ_i in $\boldsymbol{S}^{(t)}$ belongs to a pre-specified alert signature Σ. An alert signature $\Sigma = \langle \mathcal{S}, \bot, \mathcal{F}, \preceq \rangle$ is a four-tuple where $\mathcal{S}$ is

a set of sort symbols; $\mathcal{F}$ is a set of feature symbols; and $\preceq$ is a decidable partial order on $\mathcal{S}$ such that $\perp$ is the least element. Based on that order among sorts, intuitively, we say of two alerts ψ, ψ' that ψ subsumes ψ' ($\psi \sqsubseteq \psi'$) when all that is true for ψ is also true for ψ'. Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two sequences of alerts such that $\boldsymbol{X} = [\psi_1, \cdots, \psi_n]$ and $\boldsymbol{Y} = [\psi'_1, \cdots, \psi'_m]$, $|\boldsymbol{X}| = n$, $|\boldsymbol{Y}| = m$, and $n \geqslant m$. We say that $\boldsymbol{Y}$ subsumes $\boldsymbol{X}$ if there exists a sequence of indices $1 \leq i_1 < \cdots < i_m \leq n$ such that: $\psi'_1 \sqsubseteq \psi_{i_1}, \cdots, \psi'_m \sqsubseteq \psi_{i_m}$. We also define the function *root(ψ)* that returns the sort of alert ψ. A *path* $\rho(X, f_i)$ is defined as a sequence of features going from the variable X to the feature f_i. There is a *path equality* when two paths $\rho(X, f_i)$ and $\rho(Y, f_j)$ point to the same value (i.e., $\rho(X, f_i) = \rho(Y, f_j)$).

In an ever-changing environment, recalling the history of the system can be the only way to reduce uncertainty. Our model considers that as the analysis of the alert stream proceeds, it produces a probability distribution over the set of all received alerts. This probability distribution constitutes the foundation of our similarity between sequences as well as the basis that allow us to go from observations to hypotheses and from hypotheses to explanations. For each sort i in $\mathcal{S}$ we denote by $q_i^{(t)}$ the relative frequency of sort i at time t. We use the relative frequency of a sort to estimate its a priori probability $P^{(t)}(i) = q_i^{(t)}$. When there is no risk of confusion with the instant of time that we are referring to we simply use $P(i)$ and q_i. We say that the probability of occurrence of an alert ψ_j whose sort is $i = root(\psi_j)$ is $P(\psi_j) = q_{root(\psi_j)} = q_i$. Notice that given two sorts $i, j \in \mathcal{S}$ such that $i \preceq j$ then $P(i) \geqslant P(j)$ and that $P(\perp) = 1$.

We consider that our model is unable to capture all possible alerts (observable symptoms events) that affect the system under supervision. Alerts may be not evoked due to a number of causes. For example, because the corresponding network sensors cannot detect an attacker's action that corresponds to a new and unknown vulnerability. Alerts could also be lost before reaching the ACC because they are transmitted through unreliable or corrupted communication channels. We define the alert loss ratio as the probability that an alert of a given sort is lost and denote it by $L(i)$. This value is adjusted based on the knowledge about the system under supervision. For example, using the packet loss rate in the communication channel or other parameters that allow us to derive the reliability of the different components that underpin the CBR component [19].

5 Sequential Cases

A compositional (or composite) case is an assemblage of several cases that lies in a hierarchical structure. The cases on the upper levels are made up of small cases that in turn are compositional. The lowest level is made of indivisible cases. The highest level is made up of only one case that refers to the whole compositional hierarchy. Intermediate compositional cases (the cases that lie between the highest level and the lowest level) are considered as part of a larger ensemble solution rather than as individual solutions to the case at hand. We say that a case C_i is a *direct part* of a case C_j, denoted by $C_i \lhd C_j$, iff $C_i \subset C_j \wedge \nexists C_k \neq C_i : C_i \lhd C_k \wedge C_k \lhd C_j$ (i.e., they are a step away). We say that

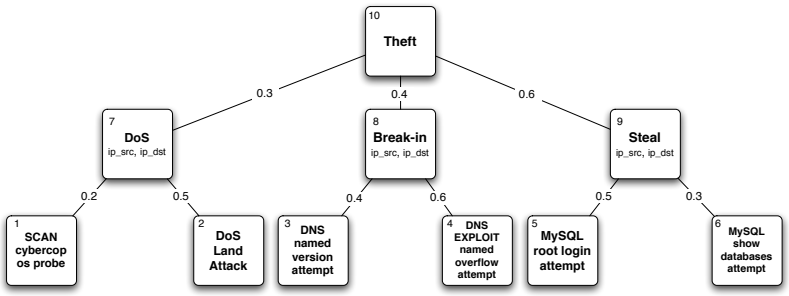

Fig. 1. An actionable tree of a Theft attack. An attacker launches a DoS against a machine running an IDS, subsequently breaks in to a DNS server, spawns a remote shell, and steals some information from a MySQL database.

case C_i is a *part* of case C_j, denoted by $C_i \lhd^* C_j$, iff there exist $n \geq 0$ cases $C_{k+1}, \cdots, C_{k+n}$ such that $C_i \lhd C_{k+1} \lhd \cdots \lhd C_{k+n} \lhd C_j$.

A sequential case is a compositional case where a temporal order is established among all the parts that comprise it. If all the sub-cases that make up a sequential case are totally-ordered then we say that the sequential case is serial. If the order of all the sub-cases are interchangeable then we say that the sequential case is parallel. Otherwise, if they are partially-ordered we say that the sequential case is arbitrary (i.e., a sequential case made up of serial and parallel subcases). Sequential cases are represented by means of *actionable trees*.

An actionable tree is a predictive compositional hierarchy modeled using a Multi-Rooted Acyclic Graph (see Fig. 1) with the semantics that a single distinguished node is designated as the *crown* (node 10), a number of (evidence) nodes are designated as the *roots* (nodes 1 to 6), the intermediate nodes in the path between the crown and a root node are designated as the *trunk* (nodes 7 to 9), and the arcs represent part-whole relationships. Roots represent observable symptom events and allow one to specify sorts to which each alert belongs. Inference is engaged upon their individual observation thus we call them roots. Nodes in the trunk represent composite (serial or parallel) cases and specify constraints (e.g. ip_src and ip_dst) among the direct parts of a sequential case. The crown represents a sequential case made up of a combination of all the events in the roots ordered according to the constraints expressed by the trunk.

A *predictive actionable tree* embodies an actionable tree in a representation that allows predictive inference. Formally, given an alert signature Σ, a predictive actionable tree is defined as a 7-tuple $\langle G, \mu, \tau, \kappa, \phi, \lhd, L \rangle$ where: G is a multi-rooted acyclic graph $G = \langle V, E \rangle$ where V is partitioned in three mutually exclusive sets: R (the set of roots) , T (the set of trunk nodes), and C (the singleton set containing the crown); E defines part-of relationships $\lhd$ among the subsets of R; $\mu : R \rightarrow \Sigma.\mathcal{S}$ is a mapping that associates each root node with a sort in $\Sigma.\mathcal{S}$; $\tau : T \cup \{C\} \rightarrow \{\mathbf{serial}, \mathbf{parallel}\}$ is a mapping that associates each non-root node with an order type; $\kappa : T \cup C \rightarrow 2^{\Sigma.\mathcal{F}}$ is a mapping that associates each non-root

node with a subset of features (constraints) in $\Sigma.\mathcal{F}$; $\phi : E \rightarrow L$ is a mapping that labels each arc $e \in E$ to a casual strength in L; and L is a likelihood model that assigns a measure of strength to the part-whole relation. L provides a probabilistic model based on the following semi-ring: $P = \langle [0,1], \cdot, 1 - \prod_{i=1}^{k}(1 - l_i), 0, 1\rangle$ where the multiplicative operation is the product of probabilities (i.e., $\cdot$) and the additive operation is defined as: $l_1 + \cdots + l_k = 1 - \prod_{i=1}^{k}(1 - l_i)$. Therefore, the probability of the crown C given a root node $r \in R$ is: $P(C|r) = \prod_{e \in path(r,C)} \phi(e)$ and the probability of the crown C given a sequence of root nodes $r_1, \cdots, r_n \in R$ is: $P(C|r_1, \cdots, r_n) = 1 - \prod_{i=1}^{n}(1 - \prod_{e \in path(r_i,C)} \phi(e))$. Actionable trees have been devised with the main purpose in mind of providing a measure of confidence on the occurrence of a whole sequence of alerts given a number of observed alerts. Said differently, they provide a measure of the strength with which the existence of the whole can be determined in terms of the existence of some of its parts. For example, consider the predictive actionable tree of Fig. 1. If we observe an alert of sort SCAN cybercop os probe then the probability of observing a Theft attack is $0.06 = 0.2 \cdot 0.3$. If we additionally observe an alert of sort DNS named version attempt then the probability of Theft is $0.2104 = 1 - ((1 - 0.06)(1 - 0.16))$.

Additionaly, we consider *sequential abstract cases*. Given an alert signature Σ, a sequential abstract case is constructed based on the informational order provided by the taxonomic hierarchy (i.e., $\Sigma.\mathcal{S}$) used to represent the alerts at the roots of the actionable tree. Sequential abstract cases allow Ceaseless CBR to find explanations for those alerts corresponding to attacks that have never occurred in the system before. That is, they are used as a back-up for the explanation of unknown situations. We use the predicate $\mathbf{abst}?(C_i)$ to determine whether a case C_i is abstract or not.

6 Case Activations

We deal with new cases that are not assembled but broken up into pieces that arrive over time without fixed boundaries and mixed in with other cases' pieces that correspond to problems that occur coincidentally. Therefore, an incremental case acquisition process is required. This process entails piecing together different parts that resemble a past case. While this happens our model needs to keep a number of plausible hypotheses that continuously best match the different partial descriptions received during a specified time span. These hypotheses, that we have called *case activations*, are generated by retrieving similar cases from the case base and are constantly updated as soon as new evidence is gathered. A case activation is a hypothesis on the occurrence of a similar past case and is represented formally as a 6-tuple $h = \langle C, \hat{a}, \check{a}, \varrho, \tilde{e}, t\rangle$ where: C is a reference to the sequential case being activated; $\hat{a}$ represents a partial binding between the sequence of alerts that occurred and are subsumed by C; $\check{a}$ represents those alerts in C that have not been observed yet and that were abduced in the dynamic sequence similarity computation; ϱ represents the rareness of the observed alerts. We compute it as the sequence similarity between C and ϱ i.e., $\varrho = C \sim_s \hat{a}$; $\tilde{e}$ measures the level of confidence (evidence) we have in the occurrence of a

complete similar sequence of alerts to those that the sequential case C represents. We compute it as the normalized sequence similarity between C and $\hat{a}$ i.e., $\tilde{e} = \|C \sim_s \hat{a}\|$; and t is the time at which the last alert on $\hat{a}$ occurred.

We define an *equality path checking* process that ensures when two case activations are compounded together the constraints established by the corresponding sequential case are followed. We say that a sequence of alerts $\boldsymbol{S}$ is constrainable given a set of constraints $\mathcal{C} = \{f_1, \cdots, f_m\} : f_i \in \Sigma.\mathcal{F}$ when path equality is kept for all features in $\mathcal{C}$ and for all alerts in $\boldsymbol{S}$. This process guarantees that all the alerts in a given sequence share a number of common features. For example, the same source and destination IP address. This process is part of the *fusion* of case activations. The fusion of two case activations $h_i = \langle C_i, \hat{a}_i, \check{a}_i, \varrho_i, \tilde{e}_i, t_i \rangle$ and $h_j = \langle C_j, \hat{a}_j, \check{a}_j, \varrho_j, \tilde{e}_j, t_j \rangle$, denoted by $h_i \uplus h_j$, is defined as $\langle C_i, \hat{a}_i \cup \hat{a}_j, \check{a}_i - \hat{a}_j, C_i \sim_s (\hat{a}_i \cup \hat{a}_j), \|C_i \sim_s (\hat{a}_i \cup \hat{a}_j)\|, \mathbf{max}(t_i, t_j) \rangle$ if h_i and h_j are *compoundable* and as $\{\langle C_i, \hat{a}_i, \check{a}_i, \varrho_i, \tilde{e}_i, t_i \rangle, \langle C_j, \hat{a}_j, \check{a}_j, \varrho_j, \tilde{e}_j, t_j \rangle\}$ otherwise. We say that two case activations $h_i = \langle C_i, \hat{a}_i, \check{a}_i, \varrho_i, \tilde{e}_i, t_i \rangle$ and $h_j = \langle C_j, \hat{a}_j, \check{a}_j, \varrho_j, \tilde{e}_j, t_j \rangle$ are compoundable when: (i) the corresponding sequential cases do not subsume repeated alerts. That is, the observed alerts in both case activations do not intersect (i.e., $\hat{a}_i \cap \hat{a}_j = \emptyset$); (ii) the observed alerts are constrainable according to the constraints expressed by the corresponding sequential case; and either (iii.a) both case activations correspond to the same sequential case, i.e., $C_i = C_j$; (iii.b) or one of the case activations corresponds to a new abstract case, i.e., $(\mathbf{abst}?(C_i) \wedge \neg\mathbf{abst}?(C_j)) \vee (\neg\mathbf{abst}?(C_i) \wedge \mathbf{abst}?(C_j))$; or both case activations correspond to a new abstract case and there exists a sequential case that can be abstracted to subsume the corresponding composition, i.e., $\mathbf{abst}?(C_i) \wedge \mathbf{abst}?(C_j) \wedge \exists C_k \in C^{(t)} : C_k \sqsubseteq \hat{a}_i \cup \hat{a}_j$. The above definition can be easily extended to the union of n case activations [1].

7 Ceaseless Retrieve

Ceaseless Retrieve continuously compares the sequence of alerts at hand with sequential cases in the case base and keeps updated a collection of case activations that represent the current *situation*. Ceaseless Retrieve proceeds as sketched by Algorithm 1. We assume a case base initially composed of $n \geqslant 0$ sequential cases $\mathbf{C}^{(0)} = \{C_1, \cdots, C_n\}$. We use $\boldsymbol{W}_{wm}^{(t)}$ to represent the most recently received alerts according to a specific window model wm. A window model determines how much context is considered each time that inference is invoked upon the arrival of new events. Time-based or space-based sliding windows are common window models. $\mathbf{H}^{(t)}$ denotes the set of current case activations. Initially $\mathbf{H}^{(0)} = \emptyset$. $\mathbf{A}^{(t)}$ denotes the set of all new case activations at iteration t. It is set to $\emptyset$ at the beginning of each iteration (line: 3). Ceaseless Retrieve establishes a case retrieval policy based on the frequency of occurrence of alerts. This policy promotes rareness. Those cases that subsume alerts that are very common receive a low score whereas those cases that subsume rare alerts receive a high score. Namely, the rarer the alerts that comprise an attack the higher the score. This helps our system to note those situations that apparently convey more peril

since the system is less used to dealing with them. The match is carried out using the dynamic sequence similarity measure that we introduced elsewhere and behaves according to such policy [1]. We denote by $\mathbf{R}^{(t)}$ the set of sequential cases retrieved at iteration t (line: 4). Using the sequence of sorts returned by $root(\boldsymbol{W}_{wm}^{(t)}(S^{(t)}))$ and our dynamic similarity measure $\sim_s$, those cases that are similar to the sequence above a user-defined threshold $0 < \theta \leqslant 1$ are retrieved. A case activation $\mathbf{h}_i$ is created for each retrieved case and fused toghether with previous case activations generated during the same iteration (lines: 5–8).

Algorithm 1 Ceaseless Retrieve

```
Require: C^(0), S^(0), θ, τ, wm;
Local: H, R, A, C_i, h_i
 1: H^(0) = ∅;
 2: while true do
 3:    A^(t) = ∅;
 4:    R^(t) = retrieve(root(W_wm^(t)(S^(t))), C^(t−1), θ);
 5:    for each C_i ∈ R^(t) do
 6:       h_i = ⟨C_i, â_i, ă_i, ϱ_i, ẽ_i, t⟩;
 7:       A^(t) = A^(t) ⊎ {h_i};
 8:    end for
 9:    for each ψ_i ∈ W_wm^(t) : D^(t)(ψ_i) = ∅ do
10:       h_i = ⟨⊥, ψ_i, ∅, ϱ*, 1, t⟩;
11:       A^(t) = A^(t) ⊎ {h_i};
12:    end for
13:    H^(t) = H^(t−1) ⊎ A^(t);
14:    for each h_i ∈ H^(t) do
15:       if h_i.t − t ⩾ τ then
16:          H^(t) = H^(t) − {h_i};
17:       end if
18:    end for
19:    send( H^(t), _CEASELESSREUSE); /* non-blocking call */
20:    [H^(t), P^(t)] = recv(_CEASELESSREUSE);
21: end while
```

We denote the domain of an alert ψ_i over time by $\mathbf{D}^{(t)}(\psi_i) = \{C_j \in \mathbf{C}^{(t-1)} : root(\psi_i) \lhd^* C_j\}$. We say that an alert is *uncovered* when its domain is $\emptyset$. For each uncovered alert in $\boldsymbol{W}_{wm}^{(t)}$, a new case activation $\mathbf{h}_i$ is created using a simple actionable tree composed uniquely of the observed alert and fused with case activations created in previous iterations (lines: 9–12). The evidence of this kind of case activation is originally set to 1 and their rareness to a maximal value (i.e., $\varrho^* = \mathbf{max}\ \varrho_i, \forall \mathbf{h}_i$) so that they could promptly be prioritized.

Those case activations that have not been altered during a certain period of time (given by the paremeter τ) are filtered out from consideration (lines: 14–17). We denote by $\boldsymbol{P}^{(t)}$ the sequence of pending alerts at time t. That is, alerts that have not been prioritized yet either because they have just arrived or they were not prioritized in a previous iteration because they had a low *urgency*. We discuss this issue later on in Sec. 8. Therefore, we say that $\mathbf{H}^{(t)}$ always keeps a number of up-to-date case activations for each pending alert. We also say that, $\mathbf{H}^{(t)}$ defines the current situation that is then sent to the Ceaseless Reuse (line: 19). Ceaseless Reuse decides on which alerts to explain/prioritize first and returns those case activations and associated alerts for which it estimates that more evidence is needed before the corresponding alerts can be prioritized conveniently (line: 20).

8 Ceaseless Reuse

Ceaseless Reuse constantly searches the combination of case activations that best explains the sequence of alerts most recently received and those that did not find an explanation in previous iterations (pending alerts). Ceaseless Reuse uses a belief function to determine which case activations are susceptible of being used to prioritize the corresponding alerts. However, if this process prioritizes alerts too soon, that is, without being completely sure of the presence of a (possible) exceptional situation the number of false positives will be high and the ultimate objective (to triage the alert stream) will not be achieved. On the contrary, if it prioritizes too late and an exceptional situation is really occurring then the time to enable a prompt response is reduced. Thus we take a decision-theoretic approach that maximizes the overall utility of each complete explanation and define a measure of urgency that guides the decisions of this process over time. Algorithm 2 sketches the tasks performed by Ceaseless Reuse.

Algorithm 2 Ceaseless Reuse

Local: $\mathbf{H}$, $\mathbf{h}_i$, $\mathbf{b}$, $\mathbf{B}$, $\mathbf{E}$, $\mathbf{e}$, $\mathbf{e}^*$, $\boldsymbol{U}$,

1: **while** true **do**
2: $\quad\mathbf{H}^{(t)} = \mathbf{recv}(\_\text{CEASELESSRETRIEVE})$;
3: $\quad$**for each** $h_i^{(t)} \in \mathbf{H}^{(t)}$ **do**
4: $\quad\quad\mathbf{b}^{(t)+}(\mathbf{h}_i) = 1$;
5: $\quad\quad$**for each** $\psi_j \in \mathbf{h}_i.\check{a}$ **do**
6: $\quad\quad\quad\mathbf{b}^{(t)+}(\mathbf{h}_i) = \mathbf{b}^{(t)+}(\mathbf{h}_i) \times \big(L(\psi_j) + ((1 - L(\psi_j)) \times (1 - P(\psi_j|\mathbf{h}_i)))\big)$;
7: $\quad\quad$**end for**
8: $\quad\quad\mathbf{b}^{(t)-}(\mathbf{h}_i) = P(\mathbf{h}_i)P(\boldsymbol{P}^{(t)}|\mathbf{h}_i)$;
9: $\quad\quad\mathbf{b}^{(t)}(\mathbf{h}_i) = \mathbf{b}^{(t)+}(\mathbf{h}_i) \times \mathbf{b}^{(t)-}(\mathbf{h}_i)$;
10: $\quad$**end for**
11: $\quad[\mathbf{H}^{(t)}, \mathbf{H}_U^{(t)}, \boldsymbol{P}^{(t)}, \boldsymbol{U}^{(t)}] = \mathbf{rank}(\mathbf{H}^{(t)}, \mathbf{b}^{(t)})$;
12: $\quad\mathbf{send}([\mathbf{H}^{(t)}, \boldsymbol{P}^{(t)}], \_\text{CEASELESSRETRIEVE})$; % non-blocking call
13: $\quad\mathbf{E}^{(t)} = \{\mathbf{e}_i \subseteq \mathbf{H}_U^{(t)} : \forall_{\psi_i \in \boldsymbol{U}^{(t)}} \exists \mathbf{h}_i \in \mathbf{e}_i \ : \mathbf{h}_i.C_i \sqsubseteq \ \psi_i \wedge \nexists \mathbf{e}'' : |\mathbf{e}''| < |\mathbf{e}'| \wedge (\mathbf{e}' \cap \mathbf{e}'') \neq \emptyset\}$;
14: $\quad$**for each** $\mathbf{e}_i \in \mathbf{E}^{(t)}$ **do**
15: $\quad\quad\mathbf{B}^{(t)+}(\mathbf{e}_i) = 1$;
16: $\quad\quad$**for** $\mathbf{h}_j \in \mathbf{e}_i$ **do**
17: $\quad\quad\quad\mathbf{B}^{(t)+}(\mathbf{e}_i) = \mathbf{B}^{(t)+}(\mathbf{e}_i) \times \mathbf{b}^{(t)+}(\mathbf{h}_j)$;
18: $\quad\quad$**end for**
19: $\quad\quad\mathbf{B}^{(t)-}(\mathbf{e}_i) = P(\mathbf{e}_i)P(\boldsymbol{U}^{(t)}|\mathbf{e}_i)$;
20: $\quad\quad\mathbf{B}^{(t)}(\mathbf{e}_i) = \mathbf{B}^{(t)+}(\mathbf{e}_i) \times \mathbf{B}^{(t)-}(\mathbf{e}_i)$;
21: $\quad$**end for**
22: $\quad\mathbf{e}^{*(t)} = \mathbf{e}_i \in \mathbf{E}^{(t)} : \mathbf{B}^{(t)}(\mathbf{e}_i)$ is **maximal**;
23: $\quad\mathbf{send}([\mathbf{e}^{*(t)}, \boldsymbol{U}^{(t)}], \_\text{CEASELESSREVISE})$; % non-blocking call
24: **end while**

At each iteration Ceaseless Reuse receives a number of competing hypotheses expressed in terms of case activations that explain the current situation (line: 2). We say that a case activation $\mathbf{h}_j$ explains an alert ψ_k if the corresponding sequential case $\mathbf{h}_j.C$ subsumes ψ_k (i.e., $\mathbf{h}_j.C_j \sqsubseteq \psi_k$). For each case activation Ceaseless Reuse computes a belief function as the product of two other belief components (lines: 3–10): a negative component (that takes into account observed alerts) and a positive component (that takes into account those alerts that have not been observed yet): $\mathbf{b}^{(t)}(\mathbf{h}_i) = \mathbf{b}^{(t)+}(\mathbf{h}_i)\mathbf{b}^{(t)-}(\mathbf{h}_i)$.

On the one hand, the positive component is computed in terms of the alerts that have been abduced during the sequence similarity computation as follows: $\mathbf{b}^{(t)+}(\mathbf{h}_i) = \prod_{\psi_j \in \mathbf{h}_i.\hat{a}} \big(L(\psi_j) + ((1 - L(\psi_j))(1 - P(\psi_j|\mathbf{h}_i))\big)$. For each abduced alert we consider every possible alternative. Namely, we consider the probability that the alert is lost $L(\psi_j)$ and the probability that the alert is not lost $(1-L(\psi_j))$ but it was not observed given the sequential case corresponding to the case activation at hand $(1-P(\psi_j|\mathbf{h}_i))$. Later we will show through Eq. 2 how to compute $P(\psi_j|\mathbf{h}_i)$, the probability that the alert was in fact observed given such case activation. On the other hand, the negative belief on a case activation $\mathbf{h}_i$ is computed as the posterior probability of the case activation given the current sequence of pending alerts: $\mathbf{b}^{(t)-}(\mathbf{h}_i) = P(\mathbf{h}_i|\boldsymbol{P}^{(t)})$. Using Bayes' theorem the posterior probability can be computed as follows: $P(\mathbf{h}_i|\boldsymbol{P}^{(t)}) = \frac{P(\mathbf{h}_i)P(\boldsymbol{P}^{(t)}|\mathbf{h}_i)}{P(\boldsymbol{P}^{(t)})}$. Notice that the denominator, the probability of the sequence of pending alerts, is a constant for all case activations at the current iteration. Therefore the relative rank produced will be the same if we only use the numerator. Thus, the computation of $\mathbf{b}^{(t)-}(\mathbf{h}_i)$ can be approximated as follows: $P(\mathbf{h}_i|\boldsymbol{P}^{(t)}) \propto P(\mathbf{h}_i)P(\boldsymbol{P}^{(t)}|\mathbf{h}_i)$. The probability of a case activation $P(\mathbf{h}_i)$ represents the probability of occurrence of the associated sequential case that in turn represents the probability of occurrence of the corresponding undesired situation. This probability is computed using the inference mechanism provided by predictive actionable trees that we saw in Sec. 5 as follows:

$$P(\mathbf{h}_i) = 1 - \prod_{\psi_j \in \mathbf{h}_i.\hat{a}} \big(1 - \prod_{e \in path(\psi_j, \mathbf{h}_i.C_i)} \phi(e)\big) \tag{1}$$

The probability that we observe the sequence of alerts $\boldsymbol{P}^{(t)}$ given the occurrence of a sequential case is computed as follows: $P(\boldsymbol{P}^{(t)}|\mathbf{h}_i) = \prod_{\psi_i \in \boldsymbol{P}^{(t)}}(1 - P(\psi_i|\mathbf{h}_i))$. Using Bayes' theorem $P(\psi_i|\mathbf{h}_i) = \frac{P(\psi_i)P(\mathbf{h}_i|\psi_i)}{P(\mathbf{h}_i)}$. By actionable trees the probability of occurrence of a sequential case given an alert is $P(\mathbf{h}_i|\psi_i) = \prod_{e \in path(\psi_j, \mathbf{h}_i.C_i)} \phi(e)$. Therefore substituting we get:

$$P(\psi_i|\mathbf{h}_i) = \frac{P(\psi_i) \prod_{e \in path(\psi_j, \mathbf{h}_i.C_i)} \phi(e)}{1 - \prod_{\psi_j \in \mathbf{h}_i.\hat{a}} \big(1 - \prod_{e \in path(\psi_j, \mathbf{h}_i.C_i)} \phi(e)\big)} \tag{2}$$

Therefore $\mathbf{b}^{(t)-}(\mathbf{h}_i)$ can be approximated using Eq. 1 and Eq. 2. Notice that a belief on a case activation does not need to be computed again and again at each new iteration. That is, we only need to recompute them when their evidence varies at the current iteration (i.e., when $\mathbf{h}_i.t$ is equal to t). Once we have computed the belief on each case activation we rank them and select a number of alerts to build an overall explanation (line: 11). The motivation for not considering all the alerts at each iteration is twofold. First, to reduce the combinatorial explosion in successive steps. The larger the number of alerts considered, the longer it will take to build an overall explanation. Second, it does not make sense to consider alerts for which our belief in their corresponding sequential case is too low, since it increases the probability of making a wrong

judgment and therefore decreasing the expected utility. Different criteria could be applied to rank and select which alerts should be explained first. Our approach is to use a measure of *urgency* in the same way that it is applied to healthcare patient monitoring and in mass casualty incidents [20].

We define urgency as the degree to which an immediate prioritization is required [20]. We compute the urgency of each alert in terms of the expected utility of prioritizing it right now, using our current degree of belief on the hypotheses that explain it, versus the expected utility of waiting to do it after more evidence has been gathered. We say that urgency allows us to trade off in prioritizing an alert versus continuing computation as well as choosing among competing case activations (see [1] for further details). Thus, given the set of current case activations $\mathbf{H}^{(t)}$ and their current beliefs $\mathbf{b}^{(t)}$ the function **rank** (line: 11) partitions alerts and their corresponding case activations into those that are urgent and and those that will remain will remain waiting for further evidence. We denote by $\boldsymbol{U}^{(t)}$ the alerts that are urgent and need to be explained at the current iteration. Likewise, $\mathbf{H}_U^{(t)}$ denotes the set of case activations that explain urgent alerts and that will be used at the current iteration to compound explanations whereas $\mathbf{H}^{(t)}$ denotes the set of case activations that remains for further iterations. Both case activations that remains for further iterations and pending alerts are sent back to Ceaseless Retrieve (line: 12). Then, Ceaseless Reuse creates explanations using the set of case activations that explain urgent alerts $\mathbf{H}_U^{(t)}$ and select the explanation whose belief is maximal to propose it to the user as the most plausible explanation.

An explanation $\mathbf{e}_i$ is a subset of $\mathbf{H}_U^{(t)}$ that explains all alerts in $\boldsymbol{U}^{(t)}$. An explanation $\mathbf{e}_i$ is said to explain an alert ψ_k if it contains at least a case activation $\mathbf{h}_j$ that explains ψ_k. $\mathbf{E}^{(t)}$ represents the set of all explantions. $\mathbf{E}^{(t)}$ is computed following a parsimonious principle [11]. Based on the observation that the probability of multiple coincidental sources is low we induce the following heuristic: $\mathbf{e}'$ is not included in $\mathbf{E}^{(t)}$ if it contains a case activation that is already contained by $\mathbf{e}'' \in \mathbf{E}^{(t)}$ such that its size is smaller. Therefore those explanations that contain case activations that appear in other explanations that are already in $\mathbf{E}^{(t)}$ and whose size is lower are not contemplated (line: 13). The next step is to compute an estimation of the goodness for each explanation in $\mathbf{E}^{(t)}$ (lines: 14–21). We define $\mathbf{B}^{(t)}(\mathbf{e}_i)$ as a belief function that represents the likelihood that all cases in $\mathbf{e}_i$ have occurred and $\mathbf{e}_i$ explains all alerts in $\boldsymbol{U}^{(t)}$. $\mathbf{B}^{(t)}$ is computed using the beliefs $\mathbf{b}^{(t)}$ previously computed for each case activation $\mathbf{h}_i$. $\mathbf{B}^{(t)+}$ is based on a double component: $\mathbf{B}^{(t)}(\mathbf{e}_i) = \mathbf{B}^{(t)+}\mathbf{e}_i)\mathbf{B}^{(t)-}(\mathbf{e}_i)$.

The belief based on positive symptoms gives a degree of suitability for each explanation that is based on the intuition that when some of the expected alerts have not occurred yet it is a positive symptom that allows us to decrease our belief on the hypotheses that compound the explanation at hand: $\mathbf{B}^{(t)+}(\mathbf{e}_i) = \prod_{\mathbf{h}_i \in \mathbf{e}_i} \prod_{\psi_j \in \mathbf{h}_i.\breve{a}} \mathbf{b}^{(t)}(\mathbf{h}_i)$. The belief component based on negative symptoms determines the relative likelihoods of multiple case activations according to their posterior probabilities: $\mathbf{B}^{(t)-}(\mathbf{e}_i) = P(\mathbf{e}_i|\boldsymbol{U}^{(t)})$. By Bayes' theorem $P(\mathbf{e}_i|\boldsymbol{U}^{(t)}) = \frac{P(\mathbf{e}_i)P(\boldsymbol{U}^{(t)}|\mathbf{e}_i)}{P(\boldsymbol{U}^{(t)})}$. To rank posterior probabilities it is only necessary to compare

the joint probabilities since the normalization factor $P(\boldsymbol{U}^{(t)})$ is a constant for all competing explanations at iteration t. Therefore: $P(\mathbf{e}_i|\boldsymbol{U}^{(t)}) \propto P(\mathbf{e}_i)P(\boldsymbol{U}^{(t)}|\mathbf{e}_i)$. The a priori probability of a explanation $\mathbf{e}_i$ is given by: $P(\mathbf{e}_i) = \prod_{\mathbf{h}_i \in \mathbf{e}_i^{(t)}} P(\mathbf{h}_i)$ that can be estimated using Eq. 1 as follows:

$$P(\mathbf{e}_i) = \prod_{\mathbf{h}_i \in \mathbf{e}_i} \left(1 - \prod_{\psi_j \in \mathbf{h}_i.\hat{a}} \left(1 - \prod_{e \in path(\psi_j, \mathbf{h}_i.C_i)} \phi(e)\right)\right) \tag{3}$$

The conditional probability of $\boldsymbol{U}^{(t)}$ given $\mathbf{e}_i$ is computed as follows: $P(\boldsymbol{U}^{(t)}|\mathbf{e}_i) = \prod_{\psi_i \in \boldsymbol{U}^{(t)}}(1 - \prod_{\mathbf{h}_i \in \mathbf{e}_i}(1 - P(\psi_i|\mathbf{h}_i)))$. Then, by Eq. 2, $P(\boldsymbol{U}^{(t)}|\mathbf{e}_i) =$

$$\prod_{\psi_i \in \boldsymbol{U}^{(t)}} \left(1 - \prod_{\mathbf{h}_i \in \mathbf{e}_i} \left(1 - \frac{P(\psi_i) \prod_{e \in path(\psi_j, \mathbf{h}_i.C_i)} \phi(e)}{1 - \prod_{\psi_j \in \mathbf{h}_i.\hat{a}} \left(1 - \prod_{e \in path(\psi_j, \mathbf{h}_i.C_i)} \phi(e)\right)}\right)\right) \tag{4}$$

Finally, $\mathbf{B}^{(t)-}$ can be approximated from Eq. 3 and Eq. 4. All explanations $\mathbf{e}_i$ in $\mathbf{E}^{(t)}$ are ranked according to $\mathbf{B}^{(t)}$. The best explanation $\mathbf{e}^{*(t)}$, the one that is maximal, among all competing explanations, is chosen as the problem solution and sent to Ceaseless Revise for user's revision (line: 23). Ceaseless Revise continuously provides a human operator with the set of most likely explanations given the alerts received so far (instead of presenting a solution periodically). The operator's feedback produces a set of revised solutions that are used by Ceaseless Revise to produce the prioritization of the corresponding alerts.

9 Conclusions

CBR practitioners are sometimes oblivious that there often situations in the real world where problems occur simultaneously and whose descriptions come interleaved or in continuous form—i.e., without well-defined boundaries between adjacent problem descriptions—and additionally require continuous response to changing circumstances—i.e., a timely action once a proper solution has been identified. In this article we have proposed to enhance the CBR paradigm to support the analysis of unsegmented sequences of observational data stemming from multiple coincidental sources. We aimed at establishing a first CBR model, that we have called Ceaseless CBR, to solve situations that are expressed by means of unsegmented, noisy sequences of complex events that arrive continuously over time. We provided a model that considers the CBR task as on-going rather than one-shot and enables reasoning in terms of problem descriptions that are broken up into small pieces that are mixed with other problems' pieces and the possibility of combining a number of cases that best match the sequential structure of the problem at hand. To put the whole matter in a nutshell, we coped here with problems that include an additional challenge compared to most of those solved by CBR practitioners before, given that each problem description is composed of an undetermined number of parts that arrive continuously over time and are blurred together with other problems' parts into a single on-line stream.

Acknowledgments We wish to acknowledge anonymous reviewers for useful suggestions and valuable corrections.

References

1. Martin, F.J.: Case-Based Sequence Analysis in Dynamic, Imprecise, and Adversarial Domains. PhD thesis, Technical University of Catalonia (2004)
2. Debar, H., Wespi, A.: Aggregation and correlation of intrusion detection alerts. In: Proceedings of the 4th Symposium on RAID. (2001)
3. Roesch, M.: Snort - lightweight intrusion detection for networks. In: Proceedings of 13th Systems Administration Conference. (1999)
4. Swets, J.A.: Signal Detection Theory and ROC Analysis in Psychology and Diagnostics. Collected Papers. Lawrence Erlbaum Associates (1996)
5. Barletta, R., Mark, W.: Breaking cases into pieces. In: AAAI-88 Case-Based Reasoning Workshop. (1988) 12–16
6. Shavlik, J.W.: Case-based reasoning with noisy case boundaries: An application in molecular biology. Technical Report 988, University of Wisconsin (1990)
7. Redmond, M.: Distributed cases for case-based reasoning; facilitating use of multiple cases. In: Proceedings of AAAI-90, AAAI Press/MIT Press (1990)
8. Ram, A., Santamaría, J.C.: Continuous case-based reasoning. In: Proceedings of the AAAI-93 Workshop on Case-Based Reasoning. (1993) 86–93
9. Jacynski, M.: A framewok for the management of past experiences with time-extended situations. In: 6th ACM CIKM. (1997)
10. Jaere, M.D., Aamodt, A., Skaalle, P.: Representing temporal knowledge for case-based prediction. In: 6th European Conference in Case-Based Reasoning. Lecture Notes in Artificial Intelligence, LNAI 2416. Springer (2002) 174–188
11. Peng, Y., Reggia, J.A.: Abductive Inference Models for Diagnostic Problem-Solving. Springer-Verlag (1990)
12. Lewis, L.: Managing Computer Networks. A Case-Based Reasoning Approach. Artech House Publishers (1995)
13. Gupta, K.M.: Knowledge-based system for troubleshooting complex equipment. International Journal of Information and Computing Science **1** (1998) 29–41
14. Breese, J.S., Heckerman, D.: Decision theoretic case-based reasoning. Technical Report MSR-TR-95-03, Microsoft Research, Advanced Technology Division (1995)
15. Aha, D.W., Maney, T., Breslow, L.A.: Supporting dialogue inferencing in conversational case-based reasoning. LNCS **1488** (1998) 262–266
16. Cunningham, P., Smyth, B.: A comparison of model-based and incremental case-based approaches to electronic fault diagnosis. In: Proceedings of the Case-Based Reasoning Workshop, AAAI-1994. (1994)
17. Emaili, M., Safavi-Naini, R., Balachandran, B., Pierprzyk, J.: Case-based reasoning for intrusion detection. In: 12th Annual Computer Security Applications Conference. (1996)
18. Schwartz, D., Stoecklin, S., Yilmaz, E.: A case-based approach to network intrusion detection. In: 5th International Conference on Information Fusion, IF'02, Annapolis, MD, July 7-11. (2002) 1084–1089
19. Steinder, M., Sethi, A.S.: Probabilistic event-driven fault diagnosis through incremental hypothesis updating. In: Proceedings of IFIP/IEEE Symposium on Integrated Network Management. (2003)
20. Huang, C., Schachter, R.: Alarms for monitoring: A decision-theoretic framework. Technical Report SMI-97-0664, Section on Medical Informatics, Stanford University School of Medicine (1997)