

# On the Formal Modelling of Trust in Reputation-Based Systems

Mogens Nielsen<sup>1,2</sup> and Karl Krukow<sup>1,2</sup>

<sup>1</sup> BRICS\*, University of Aarhus, Denmark  
{krukow, mn}@brics.dk

<sup>2</sup> Supported by SECURE \*\*

**Abstract.** In a reputation-based trust management system an entity's behaviour determines its reputation which in turn affects other entities interaction with it. We present a mathematical model for trust aimed at global computing environments which, as opposed to many traditional trust management systems, supports the dynamics of reputation-based systems in the sense that trusting relationships are monitored and changes over time depending on the behaviour of the entities involved. The main contribution is the discovery that the notion of event structures, well studied e.g. in the theory of concurrency, can faithfully model the important concepts of *observation* and *outcome* of interactions. In this setting observations are events and an outcome of an interaction is a maximal set of consistent events describing what happened. We also touch upon the problem of transferring trust or behavioural information between contexts, and we propose a generalised definition of morphism of event structures as an information-transfer function.

## 1 Introduction

In the Global Computing (GC) vision very large numbers of networked, mobile, computational entities interact to fulfill their respective goals. To be successful in such environments, entities (the terms principal, agent and entity are used synonymously) must collaborate, must be capable of operating under only partial information, and security decisions must be made autonomously, as no central authority is feasible.

The classical trust management approach [1], first introduced by Blaze, Feigenbaum and Lacy in [2], was proposed as a solution to the inadequacy of traditional security mechanisms in larger decentralised environments. Roughly, a classical trust management system deals with deciding the so-called compliance checking problem: given a request together with a set of credentials, does the request comply with the local security policy of the provider? The same authors also

---

\* Basic Research in Computer Science funded by the Danish National Research Foundation

\*\* Authors are supported by SECURE: Secure Environments for Collaboration among Ubiquitous Roaming Entities, EU FET-GC IST-2001-32486 <http://secure.dsg.cs.tcd.ie/>

developed tool-support in the form of PolicyMaker [2,3] and later KeyNote [4] for handling the trust management problem. In his paper [5], Weeks displayed a simple mathematical framework, and showed how this framework would instantiate to various existing trust management systems, including KeyNote, SPKI [6] and some logic based systems (see [5] for details), sometimes even leading to more efficient algorithms for the compliance checking problem. The framework expresses a trust management system as a complete lattice  $(D, \leq)$  of possible *authorisations*, a set of *principal names*  $\mathcal{P}$ , and a language for specifying so-called *licenses*. The lattice elements  $d, e \in D$  express the authorisations relevant for a particular system, e.g. access-rights, and  $d \leq e$  means that  $e$  authorises at least as much as  $d$ . An *assertion* is a pair  $a = \langle p, l \rangle$  consisting of a principal  $p \in \mathcal{P}$ , the *issuer*, and a monotone function  $l : (\mathcal{P} \rightarrow D) \rightarrow D$ , called a *license*. In the simplest case  $l$  could be a constant function, say  $d_0$ , meaning that  $p$  authorises  $d_0$ . In the general case the interpretation of  $a$  is: given that all principals authorise as specified in the *authorisation map*,  $m : \mathcal{P} \rightarrow D$ , then  $p$  authorises as specified in  $l(m)$ . This means that a license such as  $l(m) = m(A) \vee m(B)$  expresses a policy saying “give the lub of what  $A$  says and what  $B$  says”. Weeks showed that a collection of assertions  $L = \langle p_i, l_i \rangle_{i \in I}$  gives rise to a monotone function  $L_\lambda : (\mathcal{P} \rightarrow D) \rightarrow \mathcal{P} \rightarrow D$ , with the property that a coherent authorisation map representing the authorisations of the involved principals is given by the least fixed point,  $\text{lfp } L_\lambda$ .

The ideas on trust management systems seeded a substantial amount of research in the area of security in large distributed systems, but as noted in [7], which serves as a survey on existing systems anno 2000, the current trust management solutions do not adequately deal with the dynamic aspects of trust: a trusting relationship evolves over time and requires monitoring and reevaluation. In [8,9] it was argued that while the idea of having mutually referring licenses resolved by fixed points was good, the Weeks-framework for trust would be too restrictive in GC environments. One reason is that principals often do not have sufficient information to specify precise authorisations for all other principals. In the framework this means that any unknown or only partially known principal is always assigned the bottom authorisation. The proposed solution was to have the set  $T$  of “authorisations”, here called *trust values*, equipped with *two* orderings, denoted  $\preceq$  and  $\sqsubseteq$ . Here  $\preceq$ , called the *trust ordering*, corresponds to Weeks’ way of ordering by “more privilege”, whereas  $\sqsubseteq$ , called the *information ordering*, introduces a notion of precision or information. The key idea was that the elements of the set should embody also various degrees of uncertainty, and then  $d \sqsubseteq e$  reflects that  $e$  is more precise or contains more information than  $d$ . In the simplest of cases the trust values could be just symbolic, e.g. **unknown**  $\sqsubseteq$  **low**  $\preceq$  **high**, but they might also have more structure, as will become clear in the following sections. It was shown how least fixed points with respect to the information ordering, leads to a way of distinguishing an unknown principal from a known and distrusted one.

The notion of reputation-based systems (see e.g. [10,11,12]) also addresses some of these issues. In a reputation-based system, an agent's past behaviour determines together with local security policies how other agents assign privileges to that agent, and more generally affects any decisions concerning that agent. The SECURE project [13,14] aims at providing a framework for decision-making in GC environments, based on the notion of trust. The formal model for trust deployed is that of [8,9], and a particular application defines a triple  $(T, \sqsubseteq, \preceq)$  of trust values with the two orderings. In this model, trust exists *between principals*, and so for any principals  $P$  and  $Q$  the trust that  $P$  has in  $Q$  is modelled as an element of  $T$ . As in the Weeks-framework this value is defined in terms of a license issued by  $P$  which is called  $P$ 's *trust policy*. Thus, at any given time the trust-state of the system can be described as function,  $m : \mathcal{P} \rightarrow \mathcal{P} \rightarrow T$ , where  $\mathcal{P}$  is the set of principals, and the interpretation is that  $m(P)(Q)$  describes  $P$ 's trust in  $Q$ . At any time there is a unique trust-state describing how principals trust, and this state is the  $\sqsubseteq$ -least fixed point of the monotone function induced by the collection of all licenses.

In SECURE, each principal  $P$  has its own decision making framework which is invoked when an application needs to make some decision involving another principal. The decision making framework contains three primary components: the risk engine, the trust engine, and the collaboration monitor. At the most abstract level, the collaboration monitor records the behaviour of principals with which  $P$  has interacted. This information together with a trust policy defines how  $P$  assigns trust values to any other principal. The trust information, in turn, serves as a basis for a risk analysis of any interaction. In fact, with each type of interaction with a principal, say  $Q$ , there is a finite set of possible *outcomes* of the interaction. The outcome that occurs is determined by the behaviour of  $Q$ . Each of these outcomes has an associated cost<sup>3</sup> which could be represented simply as a number, but could also be a more complex object like a probability distribution on say  $\mathbb{R}$ . Since the outcome depends on  $Q$ , the decision of how to interact is based on the trust in  $Q$ . In this set-up it is necessary that the trust value for  $Q$  carries enough information that estimation of the likelihood of each of the outcomes is possible. If this estimation is possible, one may start reasoning about risk, e.g. the *expected* cost of an interaction. The rest of this paper describes the model for trust deployed in SECURE applications.

## 2 An evidence based framework

As we discussed in the previous section the SECURE architecture brings forward the need for a formal model for trust supporting the approximation of likelihood of interaction outcomes, based on previous observations. We now propose a framework supporting this reasoning. We will use the mathematical structures

---

<sup>3</sup> The term cost should be understood more generally as cost or *benefit*. If costs are represented as non-negative numbers, one might represent benefit as negative number.

known as event structures (see [15] for an original reference and the handbook chapter [16] for an extensive reference).

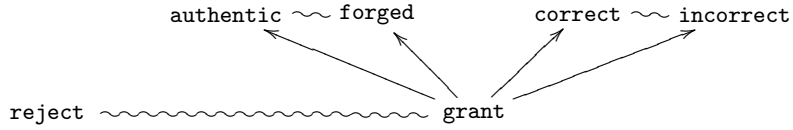
**Definition 1 (Event Structure).** An event structure is a triple  $(E, \leq, \#)$  consisting of a set  $E$  of events which are partially ordered by  $\leq$ , the necessity relation (or causality relation), and  $\#$  is a binary, symmetric, irreflexive relation  $\# \subset E \times E$ , called the conflict relation. The relations satisfy

$$\{e' \in E \mid e' \leq e\} \text{ is finite,}$$

$$\text{if } e \# e' \text{ and } e' \leq e'' \text{ then } e \# e''$$

for all  $e, e', e'' \in E$ . We say that two events are independent if they are not in either of the two relations.

As an example, the event structure in Figure 1 could model a small scenario where a principal may ask a bank for the transfer of electronic cash from its bank account to an electronic wallet. After making the request, the principal observes that the request is either rejected or granted. After a successful transaction, the principal could observe that the cash sent in the transaction is forged or perhaps run an authentication algorithm to establish that it is authentic. Also, the principal could observe a withdrawal from its bank account with the present transaction's id, and this withdrawal may or may not be of the correct amount. The two basic relations on event structures have an intuitive meaning in our set



**Fig. 1.** An event structure describing our example. The curly lines  $\sim$  describe the immediate conflict relation and pointed arrows, the causality relation.

up. An event may *exclude* the possibility of the occurrence of a number of other events. In our example the occurrence of the event 'transaction rejected' clearly excludes the event 'transaction granted'. The necessity relation is also natural: some events are *only possible* when others have already occurred. In the example structure, 'money forged' only makes sense in a transaction where the transfer of money actually did occur. Whether the e-cash is forged and whether the correct amount is charged are two independent observations that may be observed, in any order, which is modelled as independence in the event structure.

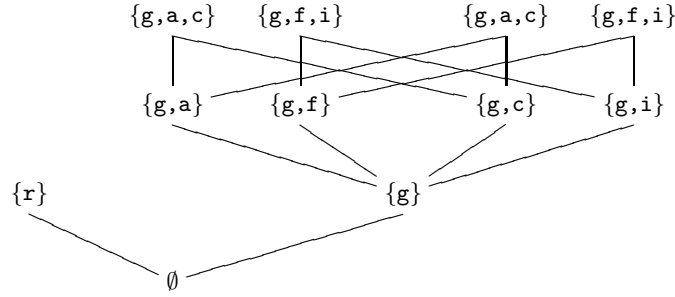
**Definition 2 (Configurations of an Event Structure).** Let  $ES = (E, \leq, \#)$  be an event structure. Say that a subset of events  $x \subseteq E$  is consistent if it satisfies the following two properties:

1. *Conflict free*: for any  $e, e' \in x : e \# e' \quad (\text{i.e. } (e, e') \notin \#)$ .
2. *Necessity downwards closed*: for any  $e \in x, e' \in E : e' \leq e \Rightarrow e' \in x$ .

Define the configurations of  $ES$ , written  $\mathcal{C}_{ES}$ , to be the set of consistent subsets of  $E$ . We will define  $\mathcal{C}_{ES}^0$  to be the finite configurations. Define relation  $\rightarrow$  on  $\mathcal{C}_{ES} \times E \times \mathcal{C}_{ES}$  by

$$x \xrightarrow{e} x' \iff e \notin x \text{ and } x' = x \cup \{e\}$$

A (finite) configuration models information regarding the result of one interaction. Note that the *outcomes* of an action corresponds to the maximal configurations (ordered by inclusion) of the event structures, and knowing the outcome corresponds to having complete information. The configurations of our example is given in Figure 2.



**Fig. 2.** Configurations of the event structure in Figure 1. The lines indicate inclusion and the events are abbreviated.

We can now be more precise about the role of the collaboration monitor in the SECURE framework. Informally, its function is to monitor the behaviour of principals with whom interaction is made. For a particular interaction the possible events that may occur are modelled by an event structure, say  $ES$ . The information about the outcome of this interaction is simply a configuration,  $x \in \mathcal{C}_{ES}^0$ .

**Definition 3 (Interaction History).** Let  $ES = (E, \leq, \#)$  be an event structure. Define an interaction history in  $ES$  to be a finite ordered sequence  $H$  of configurations,  $H = x_1 x_2 \cdots x_n \in \mathcal{C}_{ES}^{0*}$ . The individual components  $x_i$  in the history  $H$  will be called interactions.

An interaction history in the event structure from Figure 1 could be the sequence  $\{g, a, c\}\{g, c\}\{g\}\{r\}$ . The concept of interaction histories models one principal's recording of previous interactions with another. When the collaboration monitor learns about the occurrence of an event,  $e$ , this information is increased. We define a simple relation expressing this operation.

**Definition 4 (Information Relation).** Let  $ES = (E, \leq, \#)$  be an event structure and let  $H = x_1 \cdots x_n$  and  $K = y_1 \cdots y_n$  be interaction histories in  $ES$ ,  $e \in E$  an event, and  $i \in \mathbb{N}, 1 \leq i \leq n$  be an index, then define:

$$H \xrightarrow{(e,i)} K \iff x_i \xrightarrow{e} y_i \text{ and } \forall (1 \leq j \leq n) : j \neq i \Rightarrow x_j = y_j$$

and also let  $\mathbf{new}$  be a special event  $\mathbf{new} \notin E$  then

$$H \xrightarrow{\mathbf{new}} H \cdot \emptyset$$

Let  $H \Rightarrow K$  denote that either  $H \xrightarrow{\mathbf{new}} K$  or there exists  $e \in E, i \in \mathbb{N}$  so that  $H \xrightarrow{(e,i)} K$ , and  $\Rightarrow^*$  the reflexive and transitive closure of  $\Rightarrow$ .

## 2.1 Evaluating evidence

We will equate the notion of trust values with “evidence values”. That is, values expressing the amount of evidence regarding a particular partial outcome (i.e. a configuration). We will consider the derivation of such values based on interaction histories.

Consider an event structure  $ES = (E, \leq, \#)$ . A trust value will be a function from  $\mathcal{C}_{ES}$  into a domain of *evidence values*. The function applied to a configuration  $x \in \mathcal{C}_{ES}$  is then a value reflecting the evidence for  $x$ . It will be natural to express this evidence value as a triple of natural numbers  $(s, i, c) \in \mathbb{N}^3$ . The interpretation is that out of  $s + i + c$  interactions,  $s$  of these *support* the occurrence of configuration  $x$ ,  $c$  of these *contradict* it, and  $i$  are *inconclusive* about  $x$  in the sense that they do not support or contradict it.

**Definition 5.** Let  $ES = (E, \leq, \#)$  be an event structure and let  $x$  be a configuration of  $ES$ . Define the effect of  $x$  as a function,  $\mathbf{eff}_x : \mathcal{C}_{ES} \rightarrow \mathbb{N}^3$  by

$$\mathbf{eff}_x(w) = \begin{cases} (1, 0, 0) & \text{if } w \subseteq x \\ (0, 0, 1) & \text{if } x \# w \text{ (i.e. } \exists e \in x, e' \in w : e \# e') \\ (0, 1, 0) & \text{otherwise} \end{cases}$$

Also for  $(s, i, c), (s', i', c') \in \mathbb{N}^3$  define  $(s, i, c) + (s', i', c') = (s + s', i + i', c + c')$ .

The intuition behind the definition of  $\mathbf{eff}_x$  is the following. Think of  $x$  as a configuration which has already been observed. We are now considering engaging in another interaction which will end up in some configuration. Thus, we would like to estimate the likelihood of ending up in a particular configuration  $w$ , given that the last interaction ended in  $x$ . There are exactly three cases for any fixed configuration  $w$ : if  $w \subseteq x$  then the fact that  $x$  occurred last time supports the occurrence of  $w$ . If instead  $x \# w$  then  $x$  contains an event which rules out the configuration  $w$ . Finally, if neither of these are the case, i.e.  $w$  didn't occur but also wasn't excluded, we say that  $x$  is inconclusive about  $w$ . There is a strong similarity between this division of configurations in three disjoint classes and the way Jøsang [17] derives his uncertain probabilities in the Dempster-Shafer framework for evidence [18]. We discuss this in the concluding section.

**Definition 6.** Let  $ES = (E, \leq, \#)$  be an event structure, define the function  $\mathbf{eval} : \mathcal{C}_{ES}^0 \rightarrow (\mathcal{C}_{ES} \rightarrow \mathbb{N}^3)$ :

$$\mathbf{eval}(x_1 x_2 \cdots x_n) = \lambda w. \sum_{i=1}^n \mathbf{eff}_{x_i}(w)$$

We would like to note that the functions  $\mathbf{eff}$  and  $\mathbf{eval}$  allow for many useful variations when computing trust values from interaction histories. For example, suppose we want to model a "memory" so that a principal only remembers the last  $M + 1 \in \mathbb{N}$  interactions. This could be done by simply taking

$$\mathbf{eval}^M(x_1 x_2 \cdots x_n) = \lambda w. \sum_{i=n-M}^n \mathbf{eff}_{x_i}(w) = \mathbf{eval}(x_{n-M} x_{n-M+1} \cdots x_n)$$

One could also imagine older interactions "counting less", which could be modelled by scaling and rounding of the value of, say, the interactions older than a certain boundary.

## 2.2 Ordering evidence

Given this intuition we will consider two orderings on evidence values: an *information ordering*, and an ordering expressing "more evidence in favour of", which we call the *trust ordering*.

**Information order.** The information ordering  $\sqsubseteq$  of  $\mathbb{N}^3$  is defined as follows:

$$(s, i, c) \sqsubseteq (s', i', c') \iff (s \leq s') \wedge (c \leq c') \wedge (s + i + c \leq s' + i' + c')$$

The rationale is that  $(s', i', c')$  represents more information than  $(s, i, c)$  if it can be obtained from  $(s, i, c)$  by performing some additional number of interactions, or by refining the information about a particular interaction (or both). By refining we mean to change an "inconclusive" to "supporting" or "contradicting".  $\widehat{\mathbb{N}^3}$  denotes the completion of  $\mathbb{N}^3$  by a greatest element  $\top_{\sqsubseteq}$ .

**Trust order.** The trust ordering  $\preceq$  of  $\mathbb{N}^3$  is defined as:

$$(s, i, c) \preceq (s', i', c') \iff (s \leq s') \wedge (c \geq c') \wedge (s + i + c \leq s' + i' + c')$$

Here  $(s', i', c')$  expresses "more evidence in favour of" than  $(s, i, c)$  if it contains more supporting evidence, less contradicting evidence, and still at least as many interactions. Intuitively one can obtain  $(s', i', c')$  from  $(s, i, c)$  by changing contradicting evidence to inconclusive or supporting, changing inconclusive to supporting, or by adding inconclusive or positive events.

**Theorem 1.** The structure  $(\widehat{\mathbb{N}^3}, \sqsubseteq)$  is a complete lattice. The binary join is given by  $(s_0, i_0, c_0) \sqcup (s_1, i_1, c_1) = (\bar{s}, \bar{i}, \bar{c})$  where  $\bar{s} = \max\{s_0, s_1\}$ ,  $\bar{c} = \max\{c_0, c_1\}$  and

$$\bar{i} = \min\{i \in \mathbb{N} \mid \bar{s} + i + \bar{c} \geq \max\{s_0 + i_0 + c_0, s_1 + i_1 + c_1\}\}$$

The join with respect to  $\top_{\sqsubseteq}$  is as expected, and the join of any infinite set is  $\top_{\sqsubseteq}$ . Furthermore, the structure  $(\mathbb{N}^3, \preceq)$  is a lattice. The binary  $\preceq$ -join is given by  $(s_0, i_0, c_0) \vee (s_1, i_1, c_1) = (\hat{s}, \hat{i}, \hat{c})$  where  $\hat{s} = \max\{s_0, s_1\}$ ,  $\hat{c} = \min\{c_0, c_1\}$  and

$$\hat{i} = \min\{i \in \mathbb{N} \mid \hat{s} + i + \hat{c} \geq \max\{s_0 + i_0 + c_0, s_1 + i_1 + c_1\}\}$$

The meet is obtained dually. Finally, the join and meet functions for the trust order,  $\vee, \wedge : \mathbb{N}^3 \times \mathbb{N}^3 \rightarrow \mathbb{N}^3$  are monotone with respect to the information order.

In the following we use  $\sqsubseteq$  also for the pointwise extension of  $\sqsubseteq$  to trust values, i.e. the functions  $\mathcal{C}_{ES} \rightarrow \widehat{\mathbb{N}}^3$ . We can relate the relation  $\Rightarrow^*$  on interaction histories with the information relation on trust values.

**Proposition 1.** *Let  $ES$  be an event structure and  $H, K \in \mathcal{C}_{ES}^0$  interaction histories. Then **eval** is monotonic in the sense that if  $H \Rightarrow^* K$  then also  $\mathbf{eval}(H) \sqsubseteq \mathbf{eval}(K)$ .*

Some information is discarded by **eval**, and the following proposition explains what is lost. The function **eval** is injective up to rearranging the order of interactions.

**Proposition 2.** *Let  $ES = (E, \leq, \#)$  be an event structure and  $H, K \in \mathcal{C}_{ES}^0$  be configurations,  $H = x_1 x_2 \cdots x_n$  and  $K = y_1 y_2 \cdots y_m$ . If  $\mathbf{eval}(H) = \mathbf{eval}(K)$  then  $n = m$  and there exists a permutation on  $n$  elements  $\sigma : [n] \xrightarrow{\sim} [n]$  so that*

$$H = \sigma(K) = y_{\sigma(1)} y_{\sigma(2)} \cdots y_{\sigma(n)}$$

Returning to the SECURE architecture, the risk engine uses trust values to derive estimates on the likelihood of the various outcomes. Our trust values convey sufficient information to enable estimation of probability distributions on the configurations. There are several ways to do this, depending on the application. For example one might derive an opinion  $\omega_x = (b_x, u_x, d_x)$  for  $x \in \mathcal{C}_{ES}$  in the sense of Jøssang, which gives rise to a probability pdf [17,12].

### 3 Trust Policies

As discussed in the introduction, each principal defines a local *trust policy* following the idea from [5]. We give an example of a language for specifying such policies. The syntax is given in Figure 3. A policy is a list of specific policies, terminated by a general policy. The specific policies explicitly name a principal and a corresponding trust expression ( $\tau$ ), whereas the general policy applies to any principal not explicitly listed. In this simple example language, the trust expressions are built up from the basic constructs of “local reference” and “policy reference”, and these can then be combined with the various joins and meets we have available. The two types of references are similar in that both refer to a principal  $P$ 's trust value for a principal  $Q$ . The difference is that the local reference refers to  $P$ 's personal observation on  $Q$ , whereas the trust reference instead refers to the value that  $P$  would compute using its *policy*.

$$\begin{array}{ll}
\pi ::= \star : \tau & \text{(default policy)} \\
| p : \tau ; \pi & (p \in \mathcal{P}, \text{ specific policies}) \\
\tau ::= p?_{loc} q & \text{(local reference to } p, q \in \mathcal{P} \cup \{\star\}) \\
| p?q & \text{(policy reference to } p, q \in \mathcal{P} \cup \{\star\}) \\
| \tau_1 \text{ binop } \tau_2 & \text{(binary operation } \text{binop} \in \{\wedge, \vee, \sqcap, \sqcup\})
\end{array}$$

**Fig. 3.** An example policy language.

The semantics of a policy is interpreted relative to an environment providing for each pair  $P, Q$  of principals a trust value which we think of as being  $P$ 's interaction history with  $Q$  evaluated as in the previous section. This serves as the data for the local references. Let  $obs : \mathcal{P} \rightarrow \mathcal{P} \rightarrow \mathcal{C}_{ES} \rightarrow \widehat{\mathbb{N}}^3$  be a fixed function representing this. The semantics of a policy  $\pi$  is a function which takes as input the observation data  $obs$ , and gives as output a  $\sqsubseteq$ -monotone function mapping the global current trust state (an element in  $GS = \mathcal{P} \rightarrow \mathcal{P} \rightarrow (\mathcal{C}_{ES} \rightarrow \widehat{\mathbb{N}}^3)$ ), to a local trust state (an element of  $LS = \mathcal{P} \rightarrow (\mathcal{C}_{ES} \rightarrow \widehat{\mathbb{N}}^3)$ ). We denote this as

$$\llbracket \pi \rrbracket^{obs} : GS \rightarrow LS$$

The semantic function  $\llbracket \cdot \rrbracket^{obs}$  is defined by structural induction on the syntax of  $\pi$  in Figure 4. The definitions make use of the semantic function in Figure 5,

$$\begin{aligned}
\llbracket \star : \tau \rrbracket^{obs} &= \lambda m \in GS. \lambda y \in \mathcal{P}. \llbracket \tau \rrbracket^{obs}(m)([\star \mapsto y]) \\
\llbracket p : \tau ; \pi \rrbracket^{obs} &= \lambda m \in GS. \lambda x \in \mathcal{P}. \text{ if } (x = p) \text{ then } \llbracket \tau \rrbracket^{obs}(m)([\star \mapsto p]) \\
&\quad \text{else } \llbracket \pi \rrbracket^{obs}(m)(x)
\end{aligned}$$

**Fig. 4.** Semantics of the policy language: syntactic category  $\pi$ 

$$\begin{aligned}
\llbracket Y?_{loc} Z \rrbracket^{obs}(m)(env) &= obs(env^\dagger Y)(env^\dagger Z) \quad (\text{where } Y, Z \in \mathcal{P} \cup \{\star\}) \\
\llbracket Y? Z \rrbracket^{obs}(m)(env) &= m(env^\dagger Y)(env^\dagger Z) \quad (\text{where } Y, Z \in \mathcal{P} \cup \{\star\}) \\
\llbracket \tau_1 \text{ binop } \tau_2 \rrbracket^{obs}(m)(env) &= \left( \llbracket \tau_1 \rrbracket^{obs}(m)(env) \right) \llbracket \text{binop} \rrbracket \left( \llbracket \tau_2 \rrbracket^{obs}(m)(env) \right)
\end{aligned}$$

**Fig. 5.** Semantics of the policy language: syntactic category  $\tau$ 

essentially mapping the syntactic category  $\tau$  to an element of  $\mathcal{C}_{ES} \rightarrow \widehat{\mathbb{N}}^3$ . This is

again interpreted relative to observations  $obs$  and the current trust state  $m : GS$ , but also relative to an environment,  $env : \{\star\} \rightarrow \mathcal{P}$ , which interprets  $\star$  as a name in  $\mathcal{P}$ . The  $env$  function extends trivially to a function of type  $\{\star\} \cup \mathcal{P} \rightarrow \mathcal{P}$  (the identity on non- $\star$  elements). The semantics of a binop is the corresponding  $\sqsubseteq$  or  $\preceq$  lub/glb<sup>4</sup>, which is  $\sqsubseteq$ -monotone by Theorem 1.

We can now view a collection of mutually referring policies,

$$\Pi^{obs} = \{\llbracket \pi_P \rrbracket^{obs} \mid P \in \mathcal{P}\}$$

as defining a “web of trust”, and define a unique monotone function  $\Pi_\lambda^{obs}$

$$\Pi_\lambda^{obs} = \langle \llbracket \pi_P \rrbracket^{obs} : P \in \mathcal{P} \rangle : GS \rightarrow GS$$

with the property that

$$\text{Proj}_Q \circ \Pi_\lambda^{obs} = \llbracket \pi_Q \rrbracket^{obs}$$

for all  $Q \in \mathcal{P}$ . This function essentially takes a piece of global trust information  $m : GS$  and gives a piece of global trust information  $\Pi_\lambda^{obs}(m) : GS$  which, when applied to  $p$  and then to  $q$ , returns  $p$ 's trust in  $q$  under  $\pi_p$ , given trust as specified in  $m$ . Now, since the trust values  $\mathcal{C}_{ES} \rightarrow \widehat{\mathbb{N}}^3$  form a complete lattice with the information ordering, and since  $\Pi_\lambda^{obs}$  is a monotone endo-function on this structure, it has a unique least fixed point. We define the trust information in a web of trust,  $\Pi = \{\pi_p \mid p \in \mathcal{P}\}$  with local observations given by  $obs$ , as the least fixed point of the induced function,  $\llbracket \Pi \rrbracket^{obs} \stackrel{\text{(def)}}{=} \text{lfp } \Pi_\lambda^{obs}$ .

The interested reader is referred to [8,9] for examples of policies.

## 4 Transferring information

The example policy language in the previous section allows principals to share trust information by means of the reference constructs. However, we were implicitly assuming that all principals agree on the event structure used. One event structure describes a particular context, i.e. there is one event structure for each possible way of interacting. It is useful to be able to map trust values between contexts that are somehow related, e.g. if one has only very little information about context  $ES_1$  but much information about a related context  $ES_2$ , it is often useful to somehow apply the knowledge of  $ES_2$  to give an estimate in  $ES_1$ . We are aiming at formalising the kind of evidential transfer we all employ in every day life, where e.g. observations of an individual A's behaviour with respect to timely payments of bills affects also our trust in A with respect to the question of whether to lend him money. We propose a definition of a morphism of event structures enabling such an information transfer.

**Definition 7 (Morphism of event structures).** *Let  $ES = (E, \leq, \#)$  and  $ES' = (E', \leq', \#')$  be event structures. A morphism of event structure,  $\eta : ES \rightarrow ES'$  is a function  $\eta : E' \rightarrow \mathbf{2}^E$  which has the following two properties:*

<sup>4</sup> We use a strict version of the  $\preceq$ -lub/glb which is the  $\top$ -strict extension of  $\vee, \wedge : \mathbb{N}^3 \times \mathbb{N}^3 \rightarrow \mathbb{N}^3$  to a function  $\widehat{\mathbb{N}}^3 \times \widehat{\mathbb{N}}^3 \rightarrow \widehat{\mathbb{N}}^3$  which is also monotone.

1. *Monotonic:* For any  $e', e'' \in E'$  if  $e' \leq' e''$  then we have

$$\forall e_2 \in \eta(e'') \exists e_1 \in \eta(e') : e_1 \leq e_2$$

2. *Preserves conflict:* For any  $e', e'' \in E'$  if  $e' \# e''$  then

$$\forall e_1 \in \eta(e') \forall e_2 \in \eta(e'') : e_1 \# e_2$$

A morphism  $\eta : ES \rightarrow ES'$  can be thought of as a transfer of evidence from  $ES$  to  $ES'$ . The idea is that  $e \in \eta(e')$  means that an occurrence of  $e$  in  $ES$  is an indication of the event  $e'$  occurring in  $ES'$ . We will think of the set  $\eta(e')$  as a disjunction of conditions in the sense that  $e'$  occurs if there is some  $e \in \eta(e')$  which has occurred in  $ES$ . If  $\eta(e') = \emptyset$  then we say that  $e'$  has no enabling condition under  $\eta$ .

**Definition 8 (Category of Event Structures,  $\mathbf{E}$ ).** Consider the following categorical data, which we will call the category of event structures, and denote  $\mathbf{E}$ .

- Objects are event structures  $ES = (E, \leq, \#)$
- Morphisms  $\eta : ES \rightarrow ES'$ , are the morphisms of Definition 7.
- Identities  $1_{ES} : ES \rightarrow ES$  are the functions  $1_{ES} : E \rightarrow \mathbf{2}^E$  given by

$$1_{ES}(e) = \{e\}$$

- For  $\eta : ES \rightarrow ES'$  and  $\epsilon : ES' \rightarrow ES''$  composition,  $\epsilon \circ \eta : ES \rightarrow ES''$  is given by the following function  $\epsilon \circ \eta : E'' \rightarrow_* \mathbf{2}^E$

$$\epsilon \circ \eta(e'') = \bigcup_{e' \in \epsilon(e'')} \eta(e')$$

**Proposition 3 ( $\mathbf{E}$  is a category).** The definition of  $\mathbf{E}$  yields a category.

A morphism,  $\eta : ES \rightarrow ES'$  can then be used to map configurations of  $ES$  to configurations of  $ES'$  by the mapping,  $\bar{\eta} : \mathcal{C}_{ES} \rightarrow \mathcal{C}_{ES'}$

$$\bar{\eta}(x) = \{e' \in E' \mid \exists e \in \eta(e') : e \in x\}$$

The axioms of morphisms imply that  $\eta(x)$  is a configuration, and the fact that  $\mathbf{E}$  constitutes a category means that we can compose the information transfer functions to obtain information transfer functions.

## 5 Conclusion

We have proposed a mathematical framework for trust, and a way of deriving trust values from interaction histories. In this framework trust is identified with evidential information, arising from observed behaviour, allowing the estimation of likely future behaviour. The framework is deployed in the SECURE project,

and has been used in concrete SECURE prototype applications for e.g. spam filtering [14]. The trust model fits well with the bi-ordered trust structures of [8,9] and uses ideas from the framework of Weeks [5]. The way that trust values are derived from interaction histories is similar to the way that belief and plausibility functions are derived in [18], and the way in which Jøsang derives his “opinions” from belief-mass assignments in the subjective logic [17]. Event structures can be seen as a generalisation of the traditional frames of discernment from the Dempster-Shafer theory of evidence. If one allows a generalised version of event structures in which the conflict relation is allowed to be a subset of  $E \times \mathbf{2}^E$  (where  $e \# X$  means that  $e$  cannot occur if  $X$  has occurred), it is not hard to see that each frame of discernment  $\theta$  corresponds to an event structure with events  $\{\bar{p} \mid p \in \theta\}$ , where any event  $\bar{p}$  is in conflict only with the set  $\{\bar{q} \mid q \in \theta, q \neq p\}$ . The understanding of a  $\bar{p}$  is the exclusion of the state  $p$ . The configurations of the event structure is isomorphic to the poset  $(\mathbf{2}^\theta \setminus \emptyset, \subseteq)^{op}$ . Furthermore,  $x \cap y = \emptyset$  in  $\mathbf{2}^\theta \setminus \emptyset$  iff the corresponding configurations are in conflict.

While the problem of transferring trust between related contexts has been discussed, we still need to investigate the usefulness of our formalisation in terms of event structure morphisms in concrete application scenarios. The concept of morphisms seems to be appropriate for evidence transfer, but the exact definition needs some further investigation. As an example, we have considered a generalisation of the event structure morphisms presented, in which we allow  $\eta : E' \rightarrow \mathbf{2}^{C_{ES}^0}$ , i.e.  $\eta$  maps events to a disjunction of arbitrary finite configurations instead of only prime configurations (i.e. configurations of the form  $\{e \in E \mid e \leq e_0\}$  for some  $e_0$ ). This generalised definition gives rise to a category containing  $\mathbf{E}$  as a subcategory.

### Acknowledgements

We would like to thank Marco Carbone, Vladimiro Sassone and the SECURE consortium for their contribution to this work.

### References

1. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.: The role of trust management in distributed systems security. In Vitek, J., Jensen, C.D., eds.: *Secure Internet Programming*. Volume 1603 of *Lecture Notes in Computer Science*, Springer (1999)
2. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: *Proc. IEEE Conference on Security and Privacy*, Oakland. (1996)
3. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance checking in the policymaker trust management system. In: *Financial Cryptography*. (1998) 254–274
4. Blaze, M., Feigenbaum, J., Lacy, J.: KeyNote: Trust management for public-key infrastructure. *Springer LNCS* **1550** (1999) 59–63
5. Weeks, S.: Understanding trust management systems. In: *Proc. IEEE Symposium on Security and Privacy*, Oakland. (2001)
6. Ellison, C.M., Frantz, B., Lampson, B., Rivest, R., Thomsa, B., Ylonen, T.: SPKI certificate theory. *RFC* 2693 (1999)

7. Grandison, T., Sloman, M.: A survey of trust in internet application. *IEEE Communications Surveys*, Fourth Quarter (2000)
8. Carbone, M., Nielsen, M., Sassone, V.: A formal model for trust in dynamic networks. In: *Proceedings from Software Engineering and Formal Methods, SEFM'03*, IEEE Computer Society Press. (2003)
9. Krukow, K., Nielsen, M.: Towards a formal notion of trust. In: *Proceedings of the 5th ACM SIGPLAN international conference on Principles and Practice of Declarative Programming*. (2003)
10. Shmatikov, V., Talcott, C.: Reputation-based trust management. *Journal of Computer Security (selected papers of WITS '03)* (2004)
11. Mui, L., Mohtashemi, M.: Notions of reputation in multi-agent systems: A review. In: *Trust, Reputation, and Security: Theories and Practice, AAMAS 2002 International Workshop, Bologna, Italy, July 15, 2002, Selected and Invited Papers*. (2002)
12. Jøsang, A., Ismail, R.: The beta reputation system. In: *Proceedings of the 15th Bled Conference on Electronic Commerce, Bled*. (2002)
13. Cahill et al., V.: Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing* **2** (2003) 52–61
14. Cahill, V., Signeur, J.M.: Secure Environments for Collaboration among Ubiquitous Roaming Entities. Website: <http://secure.dsg.cs.tcd.ie> (2003)
15. Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures and domains. *Theoretical Computer Science* **13** (1981) 85–108
16. Winskel, G., Nielsen, M.: Models for concurrency. *Handbook of Logic in Computer Science* **4** (1995) 1–148
17. Jøsang, A.: A logic for uncertain probabilities. *Fuzziness and Knowledge-Based Systems* **9(3)** (2001)
18. Shafer, G.: *A Mathematical Theory of Evidence*. Princeton University Press (1976)