BRICS RS-03-4 Carbone et al.: A Formal Model for Trust in Dynamic Networks

BRICS

Basic Research in Computer Science

A Formal Model for Trust in Dynamic Networks

Marco Carbone
Mogens Nielsen
Vladimiro Sassone

BRICS Report Series RS-03-4
ISSN 0909-0878 January 2003

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/03/4/`

A Formal Model for Trust in Dynamic Networks

Marco Carbone[1], Mogens Nielsen[1], Vladimiro Sassone[2]

[1] BRICS*, University of Aarhus [2] COGS, University of Sussex

Abstract. We propose a formal model of trust informed by the Global Computing scenario and focusing on the aspects of trust formation, evolution, and propagation. The model is based on a novel notion of trust structures which, building on concepts from trust management and domain theory, feature at the same time a trust and an information partial order.

Introduction

Global Computing (GC) is an emerging aspect of computer science and technology. A GC system is composed of entities which are autonomous, decentralised, mobile, dynamically configurable, and capable of operating under partial information. Such systems, as e.g. the Internet, become very complex very easily, and have brought forward once again the necessity of guaranteeing security properties. Traditional security mechanisms, however, have severe limitations in this setting, as they are often either too weak to safeguard against the actual risks, or so stringent to impose unacceptable burdens on the effectiveness and flexibility of the infrastructure. *Trust management systems*, whereby safety critical decision are made based on trust policies and their deployment in the presence of partial knowledge, have an important role to play in GC.

This paper focuses on the foundations of formal models for trust in GC-like environments, capable of underpinning the use of trust-based security mechanisms as an alternative to the traditional ones.

Trust is a fundamental concept in human behaviour, and has enabled collaboration between humans and organisations for millennia. The ultimate aim of our research on trust-based systems is to transfer such forms of collaboration to modern computing scenarios. There will clearly be differences between the informal notion of trust explored in the social sciences and the kind of formality needed for computing. Mainly, our models need in the end to be operational, so as to be implementable as part of GC systems. Equally important is their role in providing a formal understanding of how trust is formed from complex interactions between individuals, so as to support reasoning about properties of trust-based systems.

Although our notion of trusted entity intends to cover only computing entities – even though of variable nature, spanning from soft to hard devices of all sorts – familiarity with trust models from the social sciences is a good starting point for our search

MC and MN supported by '**SECURE**: Secure Environments for Collaboration among Ubiquitous Roaming Entities', EU FET-GC IST-2001-32486. MC supported by '**DisCo**: Semantic Foundations of Distributed Computation', EU IHP 'Marie Curie' HPMT-CT-2001-00290. VS supported by '**MyThS**: Models and Types for Security in Mobile Distributed Systems', EU FET-GC IST-2001-32617. *Basic Research In Computer Science funded by the Danish National Research Foundation.

of a foundational, comprehensive formal model of trust. One of our main sources has been the work by McKnight and Chervany [16], who provide a typology of trust used to classify existing research on trust in domains like sociology, psychology, management, economics, and political sciences. Trust is thereby classified conceptually in six categories: *disposition*, when entity a is naturally inclined to trust; *situation*, when a trusts a particular scenario; *structure*, when a trusts impersonally the structure b is part of; *belief*, when a believes b is trustworthy; *intention*, when a is willing to depend on b; *behaviour*, when a voluntarily depends on b. Orthogonally, the notion of *trustee* is classified in categories, the most relevant of which decree that b is trusted because of its *competence*, *benevolence*, *integrity*, or *predictability*. We believe that a good mathematical model of computational trust should be capable of expressing all such aspects, as well as further notions of primary relevance in computing, e.g. that trust information is time dependent and, in general, varies very rapidly. Also, it should be sufficiently general to allow complex structures representing combinations of different types of trust.

We think of the standard deployment of a trust management system as consisting of a "*trust engine*" and a "*risk engine*" coupled together as part of a principal. The trust engine is responsible for updating trust information based on direct and indirect observations or evidence, and to provide trust information to the risk engine as input to its procedures for handling requests. The risk engine will feed back information on principals' behaviours as updating input to the trust engine. Abstracting over this point of view, we single out as central issues for our trust model the aspects of trust *formation*, *evolution*, and *propagation*. The latter is particularly important in our intended application domain, where the set of active principals is so large and open-ended that centralised trust and ad-hoc methods of propagation of its variations make absolutely no sense. An important propagation mechanism is *delegation*, whereby principals cooperate to implement complex, intertwined "global" trusting schemes. Just to pin down the idea, bank b may be willing to trust client c to an overdraft limit x only if bank b' trusts it at least up to $2x/3$, and c itself does not trust d, a crook known to b. Delegation has important consequences for trust representation, because it brings forward the idea of *trust policy*, i.e. algorithmic rules – such as bank b's above – to evaluate trust requests. In principle, trust among principals can be represented straightforwardly, as a function from pairs of principals to trust levels,

$$\mathtt{GlobalTrust} : \mathtt{Principal} \longrightarrow \mathtt{Principal} \longrightarrow \mathtt{TrustDegree}$$

where $\mathtt{GlobalTrust}(a)$ is a function which associates to each principal b the value of a's trust in b. Delegation leads to model local policies, say b's, as functions

$$\mathtt{TrustPolicy} : \mathtt{GlobalTrust} \longrightarrow \mathtt{Principal} \longrightarrow \mathtt{TrustDegree}$$

where the first argument is (a representation of) a universal trust function that b needs to know b''s level of trust in c and whether or not c trusts d.

The domain of `TrustPolicy` makes the core of the issue clear: we are now entangled in a "*web of trust*," whereby each local policy makes references to other principals' local policies. Technically, this means that policies are defined by mutual recursion, and global trust is the function determined collectively by the web of policies, the function that stitches them all together. This amounts to say that `GlobalTrust` is the least *fixpoint* of the universal set of local policies, a fact first noticed in [21] which leads straight to *domain theory* [20]. Domains are kinds of partially ordered sets which underpin the semantic theory of programming languages and have therefore been studied extensively. Working with domains allows us to use a rich and well-established theory of fixpoints to develop a theory of security policies, as well as flexible constructions to build structured trust domains out of basic ones. This is precisely context and the specific contribution of this paper, which introduces a novel domain-like structure, the *trust structures*, to assign meaning and compute trust functions in a GC scenario. We anticipate that, in due time, techniques based on such theories will find their way as part of trust engines.

As domains are partial orders (actually CPOs) and trust degrees naturally come equipped with an ordering relation (actually a lattice structure), a possible way forward is to apply the fixpoint theory to `TrustDegree` viewed as a domain. This is indeed the way of [21] and, as we motivate below, it is not a viable route for GC. There are about a million reasons in a dynamic "web of trust" why a principal a trying to delegate to b about c may not get the information it needs: b may be temporarily offline, or in the process of updating its policy, or experiencing a network delay, or perhaps unwilling to talk to a. Unfortunately, the fixpoint approach would in such cases evaluate the degree of trust of a in c to be the lowest trust level, and this decision would be wrong. It would yield the wrong semantics. Principal a should not distrust c, but accept that it has not yet had enough information to make a decision about c. What is worst with this way of confusing "trust" with "knowledge," is that the information from b could then become available a few millisecond after a's possibly wrong decision.

We counter this problem by maintaining two distinct order structures on trust values: a *trust ordering* and an *information ordering*. The former represents the degree of trustworthiness, with a least element representing, say, absolute distrust, and a greatest element representing absolute trust; the latter the degree of precision of trust information, with a least element representing no knowledge and a greatest element representing certainty. The domain-theoretic order used to compute the global trust function is the information order. Its key conceptual contribution is to introduce a notion of "*uncertainty*" in the trust value principals obtain by evaluating their policies. Its technical contribution is to provide for the "semantically right" fixpoint to be computed.

Following this lead, we introduce and study trust structures of the kind $(D, \preceq, \sqsubseteq)$, where the two order relations carry the meaning illustrated above; we then provide constructions on trust structures – including an "interval" construction which introduce a natural notion of uncertainty to complete lattices to lift them to trust structures – and use the results to interpret an toy, yet significant policy language. We believe that introducing the information ordering alongside the trust ordering is a significant step towards a model of trust feasible in a GC scenario; it is a major point of departure from the work of Weeks [21], and the central contribution of this paper.

Plan of the document. In §1 we define our trust model along the lines illustrated above, whilst §2 focuses on trust structures, providing methods for constructing useful structures as well as a general method to add uncertainty to the model. In §3 we introduce a policy language and use our trust structure to give it a denotational semantics.

Related Work. Trust is a pervasive notion and, as such, has been studied thoroughly in a variety of different fields, including social sciences, economics and philosophy. Here we only survey recent results on trust as a subject in computing. The reader is referred to [16] for a broader interpretation. A detailed survey can be found in Grandison and Sloman's [10].

Most of the existing relevant work concerns system building. In [19], Rivest *et al.* describe SDSI, a public key infrastructure featuring a decentralised name space which allows principals to create their own local names to refer to other principals' keys and in general, names. Ellison *et al.* [9] proposed a variation of the model which contributes flexible means to specify authorisation policies. The proposals are now merged in a single approach, dubbed SPKI/SDSI. Other systems of practical relevance include PGP [24], based on keys signed by trusted certificating authorities; KeyNote [3], which provides a single, unified language for both local policies and credential containing predicates to describe the trusted actions granted by (the holders of) specific public keys; PolicyMaker [2], an early version of KeyNote; and REFEREE [6], which uses a tri-valued logic which enriches the booleans with a value `unknown`. Trust in the framework of mobile agents is discussed e.g. in [22]. Delegation plays a relevant rôle in trust-based distributed systems. A classification of delegation schemes is proposed by Ding *et al.* [8], where they discuss implementation and analyse appropriate protocols. The ideas expressed in [8] lie at a different level from ours, as the focus there is exclusively on access control.

The theoretical work can be broadly divided in two main streams: logics, where the trust engine is responsible for constructing [5, 4, 12–14] or checking [1] a proof that the desired request is valid; and computational models [21, 7], like our approach.

Burrows *et al.* propose the BAN logic [5], a language for expressing properties of and reasoning about the authentication process between two entities. The language is founded on cryptographic reasoning with logical operators dealing with notions of shared keys, public keys, encrypted statements, secrets, nonce freshness and statement jurisdiction. In [4], Abadi *et al.* enhance the language by introducing delegation and groups of principals: each principal can have a particular role in particular actions. The Authorisation Specification Language (ASL) by Jajodia *et al.* [12] separates explicitly policies and basic mechanisms, so as to allow a more flexible approach to the specification and implementation of trust systems. ASL supports under a common architectural framework both the closed policy model, whereby all allowable accesses must be specified, and the open policy model, where are the denied accesses which must be explicitly specified. It also supports role-based access control.

Modal logics have a relevant place in specifying trust models, and have been used to express possibility, necessity, belief, knowledge, temporal progression, and more. Jones and Firozabadi [13] address the issue of the reliability of an agent's transmission using a modal logic of actions [17] to model agents. Rangan [18] views a distributed system as a collection of communicating agents in which an agent's state is the history of its messages, and formalises the accessibility relation which describes systems' dynamics. Rangan's model builds on simple trust statements used to defined simple properties, such as transitivity and the Euclidean property, which are then used to specify systems and analyse them with respect to properties of interest. Recently, Jøsang [14] proposed a logic of uncertain probabilities, a work which is related to our interval construction and can be recast as an instance of it in our framework. Specifically, Jøsang considers intervals of belief and disbelief over real numbers between 0 and 1.

Concerning computational models, Weeks [21] provides a model based on fixpoint computations which is of great relevance to our work. Winsborough and Li [23] study automated trust negotiation, an approach to regulate the exchange of sensitive credentials in untrusted environments. Clarke *et al.* [7] provide an algorithm for "certificate chain discovery" in SPKI/SDSI whereby principal build coherent chains of certificates to request and grant trust-based access to resources.

1 A Model for Trust

The introduction has singled out the traits of trust most relevant to our computational scenario: trust involves *entities*, has a *degree*, is based on *observations* and ultimately determines the *interaction* among entities. Our model will target these aspects primarily.

Entities will be referred to as *principals*. They form a set $\mathcal{P}$ ranged over by $a, b, c, \ldots$ and p. We assume a set $\mathcal{T}$ of *trust values* whose elements represent degrees of trust. These can be simple values, such as $\{\texttt{trusted}, \texttt{distrusted}\}$, or also structured values, e.g. pairs where the first element represents an action, say access a file, and the second a trust level associated to that action; or perhaps vectors whose elements representing benevolence in different situations.

As trust varies with experience, a model should be capable of dealing with observations resulting from the principal's interaction with the environment. For clarity, let us isolate the principal's trust management from the rest of its behaviour, and think of each principal as having a "module" containing all its trust management operations and data. Thinking "object-oriented," we can envision this as an object used by the principal for processing trust related information. Assuming a set of observations O relevant to the concrete scenario of interest, the situation could be depicted as below.

$$\texttt{request} \Longrightarrow \boxed{\begin{array}{l} \text{- }\texttt{updateTrust} : O \longrightarrow \texttt{void} \\ \text{- }\texttt{trustValue} : \mathcal{P} \longrightarrow \mathcal{T} \end{array}} \Longrightarrow t \in \mathcal{T}$$

The principal sends the message $\texttt{trustValue}(p)$ to ascertain its trust in another principal p, and the message $\texttt{updateTrust}(o)$ to add the observation o to its trust state.

In this paper, we only focus on the trust box and assume, without loss of generality, that the remaining parts of the principal interact with it using the two methods illustrated above, or some similarly suitable interface.

Modelling the Trust Box

Principals' trust in each other can be modelled as a function which associates to each pair of principals a trust value in $t \in \mathcal{T}$:

$$m : \mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}$$

Function m applied to a and then to b returns the trust value $m(a)(b) \in \mathcal{T}$ expressing a's trust in b. This however does not mean that a single principal's trust can be modelled as a function from $\mathcal{P}$ to $\mathcal{T}$, since a's trust values may depend on other principals' values. For instance, a may wish to enforce that its trust in c is b's trust in c. Similarly, we may be willing to receive a message from unknown sources, provided somebody we know trusts the sender. This mechanism of relying on third-party assessments, called *delegation*, is fundamental in all scenarios involving cooperation, including computational paradigms such as Global Computing.

This leads us to a refined view of a principal's trust as being defined by a *policy*. According to such view, each principal has a local policy π which contributes by way of delegation to form the global trust m. A policy expresses how the principal computes trust information given not just his own beliefs, but also other principals' beliefs. It follows that a's policy π_a has the type below, whose first argument represents the knowledge of third principals' policies that a needs to evaluate π_a.

$$\pi_a : (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}) \longrightarrow (\mathcal{P} \longrightarrow \mathcal{T})$$

In this paper we leave unspecified the way a policy is actually defined, as this definitely depends on the application. We study a relevant example of policy language in §3.

By collecting together the individual policies, we obtain a function $\Pi \triangleq \lambda p : \mathcal{P}.\pi_p$ whose type is (isomorphic to)

$$\Pi : (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}) \longrightarrow (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}).$$

To interpret this collection of mutually recursive local policies as a global trust function m, we apply some basic domain theory, namely fixpoints and complete partial orders. We recall below the main notions involved; in general we assume the reader to be acquainted with partial orders (cf. [11] for a thorough introduction). Given a partial order $(T, \sqsubseteq)$, an ω-chain c is a monotone function from the set of natural numbers ω to T; that is $\mathsf{c} = (c_n)_{n\in\omega}$ such that $c_0 \sqsubseteq c_1 \sqsubseteq c_2 \sqsubseteq \dots$

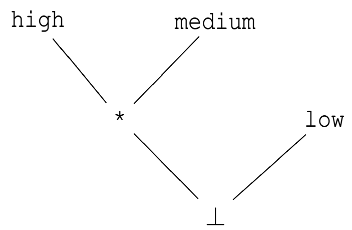
Definition 1 (CPOs and Continous functions). A partial order $(T, \sqsubseteq)$ is a *complete partial order* (CPO) if it has a least element $\perp$ and each ω-chain c in T has a least upper bound $\bigsqcup \mathsf{c}$. A function f between CPOs is continuous if for each ω-chain c, it holds that $\bigsqcup f(\mathsf{c}) = f(\bigsqcup \mathsf{c})$.

The importance of CPOs here is that every continuous function $f : (T, \sqsubseteq) \to (T, \sqsubseteq)$ on a CPO has a least fixpoint $\mathsf{fix}(f) \in T$, that is the least x such that $f(x) = x$ (cf. [20]). So, requiring $\mathcal{T}$ to be a CPO, which implies that $\mathcal{P} \to \mathcal{P} \to \mathcal{T}$ is a CPO too, and taking Π to be continuous, let us define global trust as $m \triangleq \mathsf{fix}(\Pi)$, the *least fixpoint* of Π.

The question arises however as to what order to take for $\sqsubseteq$. We maintain that it *cannot* be the order which measures the degree of trust. An example is worth many words. Let $\mathcal{T}$ be the CPO $\{\texttt{low} \le \texttt{medium} \le \texttt{high}\}$, and consider a policy π_a which delegates to b the degree of trust to assign to c. In this setup, a will assign `low` trust to c when it is not able to gather information about c from b. This however would clearly be an erroneous conclusion, as the interruption in the flow of information does not bear any final meaning about trust, its most likely cause being a transient network delay that will soon be resolved. The right conclusion for a to draw is not to distrust c, but to acknowledge that it does not know (yet) whether or not to trust c. In other words, if we want to model dynamic networks, we cannot allow confusion between “`don't trust`” and “`I don't know`:” the latter only means lack of evidence for trust or distrust, the former implies a trust-based, possibly irreversible decision.

Thus, in order to make sense of our framework, we need to introduce a notion of *uncertainty* of trust values. Truthful to the Global Computing scenario, we so account for the fact that principals may have only a partial knowledge of their surroundings and, therefore, of *their own* trust policies. We address this by considering *approximate* trust values which embody a level of uncertainty as to which value we are actually presented with. Specifically, beside the usual *trust value ordering*, we equip trust values with a *trust information ordering*. While the former measures the degree of trustworthiness, the latter measures the degree of uncertainty present in our trust information, that is its information content. We will assume that the set $\mathcal{T}$ of (approximations of) trust values is a CPO with an ordering relation $\sqsubseteq$. Then $t \sqsubseteq t'$ means that t' “refines” t, by providing more information than t about what trust value it approximates. With this understanding the continuity of Π is a very intuitive assumption: it asserts that the better determined the information from the other principals, the better determined is value returned by the policy. An example will help to fix these ideas.

Example 1. Let us refine the set of trust values $\mathcal{T}$ seen above by adding some intermediate values, viz. $\mathcal{T} = \{\perp, *, \texttt{low}, \texttt{medium}, \texttt{high}\}$, and consider the information ordering $\sqsubseteq$ specified by the following Hasse diagram:



This ordering says nothing about comparing degrees of trust. It focuses only on the quantity of information contained in values. The element $*$ represents the uncertainty as to whether `high` or `medium` holds, while $\perp$ gives no hint at all about the actual trust value. The limit of a chain reflects the finest information present in it, as a whole. Suppose we have a set of principal $\mathcal{P} = \{a, b, c\}$ with the following policies:

	a	b	c
a	`high`	$\perp$	`ask` b
b	$*$	`high`	`low`
c	`ask` b	`high`	`high`

where each row is a principal's policy. For instance the third row gives c's policy: c's trust in a is b's trust in a; c's trust in b is `high`. The computation of the least fixpoint happens by computing successive approximations following the standard theory, as illustrated below. We start from the least possible trust function:

	a	b	c
a	$\perp$	$\perp$	$\perp$
b	$\perp$	$\perp$	$\perp$
c	$\perp$	$\perp$	$\perp$

where there is absolute no knowledge and so any trust value could be anything, hence $\perp$. Each principal then inspects its own policy, and computes an approximated value using the other principals' current approximations of their values.

	a	b	c
a	`high`	$\perp$	$\perp$
b	$*$	`high`	`low`
c	$\perp$	`high`	`high`

Observe that here a is still totally undecided as to the trust to assign to c, as it knows that the value $\perp$ received from b is itself uncertain. The successive iteration is however enough to solve all solvable uncertainties, and reach the global trust function (that is the last fixpoint) illustrated below.

	a	b	c
a	`high`	$\perp$	`low`
b	$*$	`high`	`low`
c	$*$	`high`	`high`

We reiterate that, importantly, the ordering $\sqsubseteq$ is not to be identified with the equally essential ordering "more trust."

Example 2 (Collecting Observations). In order to exemplify the idea of a trust structure with two distinct orders, assume that each observation can be classified either to be *positive* or *negative*. Then let $\mathcal{T}$ be the set of pairs $\{(p,n) | p,n \in \omega + \{\infty\},\ p \leq n\}$ where p stands for the number of positive experience and n for the total number of experiences. A suitable information ordering here would be

$$(p_1,n_1) \sqsubseteq (p_2,n_2) \text{ iff } n_1 \leq n_2,$$

with $(0,0)$ being the least element. This formalises the idea that the more experience we make, the more knowledge we have. That is, trust information becomes more and more precise with interactions. On the other hand, it is intuitively clear that negative experiences cannot lead to higher trust. Therefore, a suitable trust ordering could be

$$(p_1,n_1) \preceq (p_2,n_2) \text{ iff } p_1 \leq p_2.$$

2 Trust Structures

Having pointed out the need for order structures equipped at the same time with an information and a trust ordering, in this section we focus on the triples $(\mathcal{T}, \preceq, \sqsubseteq)$, which we call *trust structures*, and study their basic properties. The notion of complete lattice, recalled below, will play a relevant role.

Definition 2 (Complete lattice). A partial order $(D, \leq)$ is a *complete lattice* if every $X \subseteq D$ has a least upper bound (lub) and, as a consequence, a greatest lower bound (glb). We use $\vee$ and $\wedge$ to denote, respectively, lubs and glbs in lattices.

When defining a trust management system, it is natural to start off with a set D of trust values, or degrees. On top of that, we are likely to need ways to compare and combine elements of D so as to form, say, a degree which comprehends a given set of trust values, or represents the trust level common to several principals. This amounts to start with a complete lattice $(D, \leq)$, where those combinators can be considered as taking lub's or glb's of sets of values. But, as illustrated above, this is not yet enough for understanding a trust framework as we need to account for uncertainty. To this purpose we define an operator I to extend a lattice $(D, \leq)$ to a trust structure $(\mathcal{T}, \preceq, \sqsubseteq)$. The set $\mathcal{T}$ consists of the set of intervals over D which, besides containing a precise image of D – viz. the singletons – represent naturally the notion of approximation, or uncertainty about elements of D.

Interval Construction

We define now the ordering $\preceq$ which has been already considered in [15].

Definition 3. Given a complete lattice $(D, \leq)$ and nonempty subsets $X, Y \subseteq D$ we say that $X \preceq Y$ if and only if

$$\wedge X \leq \wedge Y \qquad \text{and} \qquad \vee X \leq \vee Y$$

Clearly, $\preceq$ is not a partial order on the subsets of D, as the antisymmetry law fails. We get a partial order by considering as usual the equivalence classes of $\sim \, = \, \preceq \cap \succeq$. It turns out that the intervals over D are a set of representatives of such classes.

Definition 4. For $(D, \leq)$ a complete lattice, $I(D) = \{[d_0, d_1] \mid d_0, d_1 \in D, \ d_0 \leq d_1\}$, where $[d_0, d_1] = \{d \mid d_0 \leq d \leq d_1\}$ is the interval of D determined by d_0 and d_1.

Proposition 1. *Let $X = [d_0, d_1]$ be an interval in D. Then, $\wedge X$ is d_0 and $\vee X$ is d_1.*

As a consequence of the proposition above we have that $X \sim [\wedge X, \vee X]$, for all $X \subseteq D$. Furthermore, $[d_0, d_1] \sim [d_0', d_1']$ implies that $d_0 = d_0'$ and $d_1 = d_1'$.

The following lemma characterises $\preceq$ in terms of $\leq$.

Lemma 1. *For $[d_0, d_1]$ and $[d_0', d_1']$ intervals of D, we have $[d_0, d_1] \preceq [d_0', d_1']$ if and only if $d_0 \leq d_0'$ and $d_1 \leq d_1'$.*

We can now show that the lattice structure on $(D, \leq)$ is lifted to a lattice structure $(I(D), \preceq)$ on intervals.

Theorem 1. $(I(D), \preceq)$ *is a complete lattice.*

Proof. Let S be a subset of $I(D)$. We prove that its least upper bound exists. Observe that this is enough to conclude, as $\bigwedge X$ is equal to $\bigvee\{y \,|\, y \preceq x$ for all $x \in X\}$. Let S be $\{[d_0^i, d_1^i] \,|\, i \in J\}$ for some set J. We claim that $\bigvee S = [\vee d_0^i, \vee d_1^i]$.

As d_0^i and d_1^i are elements of a complete lattice $\vee d_0^i$ and $\vee d_1^i$ exist. Moreover for each j we have $d_0^j \leq d_1^j \leq \vee d_1^i$, so $\vee d_0^i \leq \vee d_1^i$ which implies $[\vee d_0^i, \vee d_1^i] \in I(D)$. Also, $[\vee d_0^i, \vee d_1^i]$ is an upper bound as for each j we have $[d_0^j, d_1^j] \preceq [\vee d_0^i, \vee d_1^i]$. Finally, $[\vee d_0^i, \vee d_1^i]$ is the least upper bound: if for each j we have that $[d_0^j, d_1^j] \preceq [d_0, d_1]$ then $[\vee d_0^i, \vee d_1^i] \preceq [d_0, d_1]$. This holds since $d_0^j \leq d_0$ and $d_1^j \leq d_1$ which means that $\vee d_0^i \leq d_0$ and $\vee d_1^i \leq d_1$. $\square$

We define an ordering on intervals which reflects their information contents so to complete the lattice structure with a CPO to base fixpoint computations on. The task is quite easy: as the interval $[d_0, d_1]$ expresses a value between d_0 and d_1, the narrower the interval, the fewer the possible values. This leads directly to the following definition.

Definition 5. For $(D, \leq)$ a complete lattice and $X, Y \in I(D)$, define $X \sqsubseteq Y$ if $Y \subseteq X$.

Analogously to $\preceq$, we can characterise $\sqsubseteq$ in terms of $\leq$. The proof follows a similar patter and is therefore omitted.

Lemma 2. *For $[d_0, d_1]$ and $[d_0', d_1']$ intervals of D, we have that $[d_0, d_1] \sqsubseteq [d_0', d_1']$ if and only if $d_0 \leq d_0'$ and $d_1' \leq d_1$.*

Finally, as for the previous ordering, we have the following result.

Theorem 2. *$(I(D), \sqsubseteq)$ is a CPO.*

Proof. The least element of $(I(D), \sqsubseteq)$ is $D = [\wedge D, \vee D]$. Let $[d_0^n, d_1^n]_n$ be an ω-chain in $(I(D), \sqsubseteq)$. Then we claim that $\bigsqcup [d_0^n, d_1^n]_n = [\vee d_0^n, \wedge d_1^n]$. We need to prove that this is well-defined and that it is the least upper bound.

As in the proof of Theorem 1, $\vee d_0^i$ and $\wedge d_1^i$ exist. Moreover, if for all i and j it holds that $d_0^i \leq d_1^j$, then for all j we have that $\vee d_0^i \leq d_1^j$, hence $\vee d_0^i \leq \wedge d_1^i$, which implies that $[\vee d_0^n, \wedge d_1^n]$ is well defined. Interval $[\vee d_0^n, \wedge d_1^n]$ is an upper bound as for all j it holds that $[d_0^j, d_1^j] \sqsubseteq [\vee d_0^i, \wedge d_1^i]$. In fact for all j we have $d_0^j \leq \vee d_0^i$ and $\wedge d_1^i \leq d_1^j$, which means $[\vee d_0^i, \wedge d_1^i] \sqsubseteq [d_0^j, d_1^j]$. Finally, $[\vee d_0^n, \wedge d_1^n]$ is the least upper bound as for any interval $[d_0, d_1]$ if $[d_0^j, d_1^j] \sqsubseteq [d_0, d_1]$ for all j then $[\vee d_0^i, \wedge d_1^i] \sqsubseteq [d_0, d_1]$. In fact $d_0^j \leq d_0$ and $d_1 \leq d_1^j$. By definition of lub and glb we have that $\vee d_0^i \leq d_0$ and $d_1 \leq \wedge d_1^i$. $\square$

The trust structures above give a constructive method to model trust based systems. We remark that intervals are a natural way to express partial information: trust in a principal is $[d_0, d_1]$ when it could be any value in between d_0 and d_1.

Example 3 (Intervals in [0,1]). Let R stand for the set of reals between 0 and 1, which is a complete lattice with the usual ordering $\leq$. We can now consider the set $I(R)$ of intervals in R. From the previous results it follows that $(I(R), \preceq)$ is a complete lattice and $(I(R), \sqsubseteq)$ is a complete partial order. Hence, if we start with a domain of trust values which are elements in R we can apply our model to the new domains. This kind of construction on reals is related to the uncertainty logic [14] where an interval $[d_0, d_1]$

in $I(R)$ is seen as a pair of numbers where d_0 is called belief and $1-d_1$ disbelief. This new trust domain is particularly interesting since it allows to express complex policies. We shall see a few examples later on.

Lifting Operators

The continuity of the function Π is an important requirement. This property depends on the operators used with the policies. In the sequel we give a useful result, wrt our interval construction, which allows to lift continuous operators in the original lattice $(D,\leq)$ to continuous operators in $(I(D),\sqsubseteq)$ and $(I(D),\preceq)$.

Definition 6. For $(D,\leq)$ and $(D',\leq')$ complete lattices and $f: D \longrightarrow D'$ a continuous function, let $I(f): I(D) \longrightarrow I(D')$ be the *pointwise lifting* of f defined as

$$I(f)([d_0,d_1]) = [f(d_0), f(d_1)].$$

In the definition above, note that the continuity of f ensures that $I(f)$ is well defined.

An ω-cochain in a complete lattice $(D,\leq)$, is an antitone function $\mathsf{c}: \omega \to T$, that is a function such that $i \leq j$ implies $c_j \leq c_i$. A function $f: (D,\leq) \longrightarrow (D',\leq')$ is co-continuous iff for each ω-cochain c in D, it holds that $\bigwedge f(\mathsf{c}) = f(\bigwedge \mathsf{c})$; f is bi-continuous if it is continuous and co-continuous.

The following proposition states that ω-cochains in $(I(D),\sqsubseteq)$ have glbs.

Proposition 2. *Let $[d_0^n, d_1^n]$ be an ω-cochain in $(I(D),\sqsubseteq)$. Then $\bigsqcap [d_0^n, d_1^n] = [\wedge d_0^n, \vee d_1^n]$.*

Proof. The proof proceeds symmetric to that of Theorem 2. □

We can now give the following result about lifted functions in trust structures.

Theorem 3. *For $(D,\leq)$ and $(D',\leq')$ complete lattices and $f: D \longrightarrow D'$ a bi-continuous function, the pointwise lifting $I(f)$ is bi-continuous with respect both the information and the trust orderings.*

Proof. We show that $I(f)$ is bi-continuous with respect to the information ordering. First we prove that $I(f)$ is continuous, i.e. $\bigsqcup I(f)([d_0^n, d_1^n]) = I(f)(\bigsqcup [d_0^n, d_1^n])$ for $[d_0^n, d_1^n]$ an ω-chain. By definition of $I(f)$ we have that $\bigsqcup I(f)([d_0^n, d_1^n]) = \bigsqcup [f(d_0^n), f(d_1^n)]$ and from the proof of Theorem 2 and bi-continuity of f it follows that

$$\bigsqcup [f(d_0^n), f(d_1^n)] = [\vee f(d_0^n), \wedge f(d_1^n)] = [f(\vee d_0^n), f(\wedge d_1^n)] = I(f)(\bigsqcup [d_0^n, d_1^n]).$$

The proof that $I(f)$ is co-continuous follows the same patterns. Let $[d_0^n, d_1^n]$ be an ω-cochain. Then, $\bigsqcap I(f)([d_0^n, d_1^n]) = \bigsqcap [f(d_0^n), f(d_1^n)]$ and from the proposition above and by the bi-continuity of f it follows that

$$\bigsqcap [f(d_0^n), f(d_1^n)] = [\wedge f(d_0^n), \vee f(d_1^n)] = [f(\wedge d_0^n), f(\vee d_1^n)] = I(f)(\bigsqcap [d_0^n, d_1^n]).$$

The proof for the trust orderings proceeds similarly. □

In the following examples we show how to apply the previous theorem to some interesting operators.

Example 4 (Lub and glb operators). The most natural operators, regarding lattices, are lub and glb. It is easy to see that they are bi-continuous in $(D, \leq)$ complete lattice. Exploiting Theorem 3 we can now state that lub and glb wrt $\preceq$ are bi-continuous over $(I(D), \sqsubseteq)$.

Example 5 (Multiplication and Sum). When considering the interval construction over R, as in Example 3, we can lift the operators of sum (weighed) and multiplication over the intervals. In fact, given two intervals $[d_0, d_1]$ and $[d'_0, d'_1]$, the product is defined as

$$[d_0, d_1] \cdot [d'_0, d'_1] = [d_0 \cdot d'_0, d_1 \cdot d'_1].$$

which is exactly the lifting of multiplication over reals. Similarly we can define sum as

$$[d_0, d_1] + [d'_0, d'_1] = [d_0 + d'_0 - d_0 \cdot d'_0, d_1 + d'_1 - d_1 \cdot d'_1]$$

These operations appears in [14] under the names of conjunction and disjunction.

Example 6 (A non-lifted operator: Discounting). Discounting, as defined in [14], is an operator which weighs the trust value received from a delegation according to the trust in the delegated principal.

$$[d_0, d_1] \rhd [d'_0, d'_1] = [d_0 \cdot d'_0, 1 - d_0 \cdot (1 - d'_1)]$$

Notice that this operator is not commutative.

2.1 Product and Function Constructors

Our model should satisfy "context dependent" trust. By this we mean that trusting a principal a to obtain information about restaurants does not mean that we trust a about, say, sailing. We can accommodate this kind of situations using a simple property of lattices and CPO's. Namely, we can form products of trust structures where each component accounts for a particular context. For instance, using a domain of the form $Restaurants \times Sailing$ will allow us to distinguish about a's dependability on the two issues of our example. The next theorem shows that extending the orders pointwise to products and function spaces gives the result we need.

Theorem 4. *Given two complete lattices $(D, \leq)$, $(D', \leq')$ and a generic set X then*

1. *$I(D \times D')$ is isomorphic to $I(D) \times I(D')$;*
2. *$X \longrightarrow I(D)$ is isomorphic to $I(X \longrightarrow D)$.*

Proof. In both cases we have to show that there exists a bijective correspondence H which preserves the orderings. We use as usual (d, d) to denote pairs and $\lambda x.t(x)$ to express the function which takes a value d and returns $t(d)$.

1. Let $[(d_0, d'_0), (d_1, d'_1)]$ and $[(d_2, d'_2), (d_3, d'_3)]$ be in $I(D \times D')$. We define the function $H : I(D \times D') \longrightarrow I(D) \times I(D')$ by

$$H([(d_0, d'_0), (d_1, d'_1)]) = ([d_0, d_1], [d'_0, d'_1])$$

which is easily seen to be bijective. We first show that H is well defined, i.e. that $d_0 \leq d_1$ and $d'_0 \leq' d'_1$. This follows at once, since $(d_0, d'_0) \leq_{D\times D'} (d_1, d'_1)$ where $\leq_{D\times D'}$ is the pointwise extension of $\leq$ and $\leq'$. Next we prove that H preserves and reflects $\preceq$. This amounts to prove that

$$[(d_0, d'_0), (d_1, d'_1)] \preceq_{I(D\times D')} [(d_2, d'_2), (d_3, d'_3)] \tag{1}$$

if and only if

$$([d_0, d_1], [d'_0, d'_1]) \preceq_{I(D)\times I(D')} ([d_2, d_3], [d'_2, d'_3]) \tag{2}$$

By Lemma 1, relation (1) above holds if and only if

$$(d_0, d'_0) \leq_{D\times D'} (d_2, d'_2) \quad \text{and} \quad (d_1, d'_1) \leq_{D\times D'} (d_3, d'_3). \tag{3}$$

Then, by the same Lemma 1, property (3) holds if and only if

$$[d_0, d_1] \preceq_{I(D)} [d_2, d_3] \quad \text{and} \quad [d'_0, d'_1] \preceq_{I(D')} [d'_2, d'_3],$$

which is the same as saying that (2) holds.
The proof is similar for the information ordering.

2. Let $[f_0, f_1]$ and $[f'_0, f'_1]$ be in $I(X \longrightarrow D)$ and g in $X \longrightarrow I(D)$. We define the bijection $H : I(X \longrightarrow D) \cong (X \longrightarrow I(D))$ by $H([f_0, f_1]) = \lambda x.[f_0(x), f_1(x)]$. It is easy to see that $H([f_0, f_1])$ is well defined. The function $H^{-1}(g) = [\lambda x. \wedge g(x), \lambda x. \vee g(x)]$ is the inverse of H. In fact, $H^{-1}(H([f_0, f_1])) = H^{-1}(\lambda x.[f_0(x), f_1(x)])$ and by definition of H^{-1} we have that the latter coincides with

$$[\lambda y. \wedge (\lambda x.[f_0(x), f_1(x)])(y), \lambda y. \vee (\lambda x.[f_0(x), f_1(x)])(y)],$$

which is $[\lambda x. \wedge [f_0(x), f_1(x)], \lambda x. \vee [f_0(x), f_1(x)]]$, i.e. $[f_0, f_1]$. Conversely, we have $H(H^{-1}(g)) = H([\lambda x. \wedge g(x), \lambda x. \vee g(x)])$ and, by definition of H, this is the same as

$$\lambda y.[(\lambda x. \wedge g(x))(y), (\lambda x. \vee g(x))(y)] = \lambda x.[\wedge g(x), \vee g(x)] = g.$$

We now need to show that H and H^{-1} preserve $\sqsubseteq$. Regarding H we have to prove

$$[f_0, f_1] \sqsubseteq_{I(X\longrightarrow D)} [f'_0, f'_1] \tag{4}$$

implies

$$\lambda x.[f_0(x), f_1(x)] \sqsubseteq_{X\longrightarrow I(D)} \lambda x.[f'_0(x), f'_1(x)] \tag{5}$$

From Lemma 2 and (4) it follows that $f_0 \sqsubseteq_{X\longrightarrow D} f'_0$ and $f'_1 \sqsubseteq_{X\longrightarrow D} f_1$ and, as the ordering is pointwise, we have that for all x, $f_0(x) \leq f'_0(x)$ and $f'_1(x) \leq f_1(x)$. Again by Lemma 2, we obtain $[f_0(x), f_1(x)] \sqsubseteq [f'_0(x), f'_1(x)]$ which, by pointwise ordering, implies (5).
Regarding H^{-1}, we show that

$$g \sqsubseteq_{X\longrightarrow I(D)} g' \tag{6}$$

implies

$$[\lambda x. \wedge g(x), \lambda x. \vee g(x)] \sqsubseteq_{I(X\longrightarrow D)} [\lambda x. \wedge g'(x), \lambda x. \vee g'(x)] \tag{7}$$

From (6) we have that, for any x, $[\wedge g(x), \vee g(x)] \sqsubseteq [\wedge g'(x), \vee g'(x)]$. It then follows that $\wedge g(x) \leq \wedge g'(x)$ and $\vee g'(x) \leq \vee g(x)$ which implies $\lambda x. \wedge g(x) \leq \lambda x. \wedge g'(x)$ and $\lambda x. \vee g'(x) \leq \lambda x. \vee g(x)$. Finally, again by Lemma 2, we have that (7) holds.

The proof for the trust ordering is similar and, thus, omitted. $\square$

Remark 1. Theorem 3 holds for any bi-continuous function $f : D_0 \times \ldots \times D_n \longrightarrow D$. The pointwise lifting of f gives $I(f) : I(D_0 \times \ldots \times D_n) \longrightarrow I(D)$ and from the result above we have that $I(f)$ is (isomorphic to) a function $F : I(D_0) \times \ldots \times I(D_n) \longrightarrow I(D)$.

3 A Policy Language

Following our discussion we propose to operate with a language for trust policies capable of expressing intervals, delegation, and a set of function constructions. We exemplify the approach by studying the simple policy language below

Syntax

The language consists of the following syntactic categories, parametric over a fixed trust lattice $(D, \leq)$.

$$
\begin{array}{llr@{\quad}lllr}
\pi ::= & \ulcorner p \urcorner & \text{(delegation)} & p ::= & a \in \mathcal{P} & & \text{(principal)} \\
 & \mid \lambda x : \mathcal{P}.\ \tau & \text{(abstraction)} & & \mid x : \mathcal{P} & & \text{(vars)} \\
 & & & & & & \\
\tau ::= & [d, d] \in I(D) & \text{(value/var)} & e ::= & \iota \ \mathtt{eq} \ \iota & \iota \in \{\tau, p\} & \text{(equality)} \\
 & \mid \pi(p) & \text{(policy value)} & & \mid \tau \ \mathtt{cmp} \ \tau & & \text{(comparison)} \\
 & \mid e \mapsto \tau; \tau & \text{(choice)} & & \mid e \ \mathtt{bop} \ e & & \text{(boolean op)} \\
 & \mid \mathtt{op}(\tau_1 \ldots \tau_n) & \text{(lattice op)} & & & &
\end{array}
$$

Here $\mathtt{op}$ is a continuous function over $(I(D), \sqsubseteq)$; operator $\mathtt{eq}$ is equality in $\tau \cup p$, $\mathtt{cmp}$ is one of $\sqsubseteq$ and $\preceq$, and $\mathtt{bop}$ is a standard boolean operator. The elements of the category p are either principals or variables.

The main syntactic category is π: it can be either delegation to another principal or a λ-abstraction. An element of τ can be an interval, the application of a policy, a conditional or the application of a continuous operator $\mathtt{op}$. The elements of e are boolean functions applied to elements of $\mathcal{P}$, D and $I(D)$.

Semantics

We provide a formal semantics for the language described above. As pointed out before, π is a policy. Hence the semantic domain, as described in §1, will be the codomain of the function

$$[\![\pi]\!]_\sigma : \ (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}) \longrightarrow (\mathcal{P} \longrightarrow \mathcal{T}),$$

where σ is an assignment of values in $\mathcal{P}$ to variables. The semantic function $[\![\cdot]\!]_\sigma$ is defined by structural induction on the syntax of π as follows.

$$[\![\ulcorner p \urcorner]\!]_{\sigma m} = m(\![p]\!)_{\sigma m};$$
$$[\![\lambda x : P.\ \tau]\!]_{\sigma m} = \lambda p : \mathcal{P}.(\![\tau]\!)_{\sigma\{p/x\}m}.$$

Here $(\![\cdot]\!)_{\sigma m}$ is a(n overloaded) function which given an assignment σ and a global trust function $m : \mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}$ maps elements of p, τ, and e respectively to the semantic domains $\mathcal{P}$, $I(D)$, and Bool as follows.

$$\begin{aligned}
(\![\ [d_0, d_1]\ ]\!)_{\sigma m} &= [(\![d_0]\!)_{\sigma m}, (\![d_1]\!)_{\sigma m}] \\
(\![\pi(p)]\!)_{\sigma m} &= [\![\pi]\!]_{\sigma m}([\![p]\!]_{\sigma m}) \\
(\![e \mapsto \tau_1; \tau_2]\!)_{\sigma m} &= \texttt{if}\ (\![e]\!)_{\sigma m}\ \texttt{then}\ (\![\tau_1]\!)_{\sigma m}\ \texttt{else}\ (\![\tau_2]\!)_{\sigma m} \\
(\![\texttt{op}(\tau_1 \ldots \tau_n)]\!)_{\sigma m} &= op\,((\![\tau_1]\!)_{\sigma m}, \ldots, (\![\tau_n]\!)_{\sigma m}) \\
(\![a]\!)_{\sigma m} &= a \\
(\![x]\!)_{\sigma m} &= \sigma(x)
\end{aligned}$$

We omit the rules for the syntactic category e, as they are obvious.

Let $\{\pi_p\}_{p \in \mathcal{P}}$ be a an arbitrary collection of all policies, where $\pi_p = \lambda x : \mathcal{P}.\ \perp$ for all but a finite number of principals. The fixpoint semantics of $\{\pi_p\}_{p \in \mathcal{P}}$ is the global trust function determined by the collection of individual policies, and it is readily expressed in terms of $[\![\cdot]\!]_\sigma$:

$$\{\![\ \{\pi_p\}_{p \in \mathcal{P}}\ ]\!\}_\sigma = \mathsf{fix}(\lambda m.\lambda p.[\![\pi_p]\!]_{\sigma m})$$

We believe that this policy language is sufficiently expressive for most application scenarios in global computing, as illustrated by our examples which follows. Note however that our approach easily generalises to any choice of underlying trust structure $(\mathcal{T}, \preceq, \sqsubseteq)$. The only requirement is that the operators used in the policy language are continuous with respect to the information ordering

Example 7 (Read and Write access). Let $D = \{\texttt{N}, \texttt{W}, \texttt{R}, \texttt{RW}\}$ represent the access rights to principal's CVs. The set D is ordered by the relation $\leq$

$$\forall d \in D.\texttt{N} \leq d \qquad \text{and} \qquad \forall d \in D.d \leq \texttt{RW}$$

We now show how to express some simple of policies in our language. For instance the following policy says that `LIZ`'s trust in `BOB` is at least `[W,RW]` and depends on what she thinks of `CARL`. Instead, `LIZ`'s trust in `CARL` will depend on her trust in `BOB`: if it is above `[W,W]` then `[R,RW]` otherwise `[N,RW]`.

$$\begin{aligned}
\pi_{\texttt{LIZ}} = \lambda x : \mathcal{P}.\ & x = \texttt{BOB} \mapsto \texttt{[W,RW]} \vee \ulcorner\texttt{LIZ}\urcorner(\texttt{CARL}); \\
& x = \texttt{CARL} \mapsto \big(\texttt{[W,W]} \preceq \ulcorner\texttt{LIZ}\urcorner(\texttt{BOB}) \mapsto \texttt{[R,RW]}; \texttt{[N,RW]}\big); \\
& \texttt{[N,RW]}
\end{aligned}$$

We could also extend the previous policy making it dependent on someone else's belief. For instance, in the following policy, the previous judgement wrt BOB has been merged with PAUL's belief (weighed by discounting).

$$\begin{aligned}\pi_{\texttt{LIZ}} = \lambda x : \mathcal{P}.\ & x = \texttt{BOB} \mapsto \texttt{[N,W]} \vee \ulcorner\texttt{LIZ}\urcorner(\texttt{CARL}) \vee \ulcorner\texttt{LIZ}\urcorner(\texttt{PAUL}) \rhd \ulcorner\texttt{PAUL}\urcorner(x);\\ & x = \texttt{CARL} \mapsto (\,\texttt{[W,W]} \preceq \ulcorner\texttt{LIZ}\urcorner(\texttt{BOB}) \mapsto \texttt{[R,RW]}\,;\texttt{[N,RW]}\,);\\ & \texttt{[N,RW]}\end{aligned}$$

In this case LIZ's trust in PAUL is the bottom value [N,RW] which is going to be the left argument of the discounting operator $\rhd$.

Example 8 (Spam Filter). Let R be the subset of real numbers between 0 and 1 ordered by the usual $\leq$, as in Example 3. We can now show some policies modelling spam filters for blocking certain emails considered spam. The set of principals $\mathcal{P}$ is the set of internet domains from which we could receive emails, e.g. daimi.au.dk. A starting policy, where we suppose that our server spam.filter.edu knows no one, could be the following

$$\pi_1 = \lambda x : \mathcal{P}.\ x = \texttt{spam.filter.edu} \mapsto [1,1]; [0,1],$$

meaning that only internal emails are trusted. It could happen that spam.filter.edu starts interacting with other principals. A likely event is that it receives a list of other university internet domains, and decides to almost fully trust them and actually use their beliefs. Then we would have

$$\pi_2 = \lambda x : \mathcal{P}.\ x \in \mathit{UniList} \mapsto [.75,1];\ \bigvee_{y \in \mathit{Unilist}} \ulcorner y \urcorner(x) \vee \pi_1(x),$$

where we suppose that "$\in$" stands for a chain of nested conditionals for all the elements of *UniList*. Let us suppose now that the filter receives emails, judged bad from a certain number of addresses, and would like to be aware of it in order to take further decisions. Then the policy would be updated to the following

$$\pi_3 = \lambda x : \mathcal{P}.\ x \in \mathit{BadList} \mapsto [0,.5];\ \pi_2(x).$$

At a certain stage the spam-filter could decide to add a new level of badness and create the new list *VeryBadList*. But at the same time it would like to change the policy with respect to *BadList* putting certain restrictions on the intervals which take care of other universities' opinions.

$$\pi_4 = \lambda x : \mathcal{P}.\ x \in \mathit{VeryBadList} \mapsto [0,.2];\ x \in \mathit{BadList} \mapsto \pi_2(x) \wedge_{\mathrm{op}} [0,.5]; \pi_2(x).$$

As illustrated in the *Spam Filter* example, we see trust evolution as being modelled by suitable updates of policies, as response to e.g. observations of principal behaviour. However, it is not clear exactly what update primitives are required in practice. We are currently working on developing a calculus of principal behaviour, with features for trust policy updates, i.e. instantiating the function update introduced in §1.

Conclusion

We presented a novel model for trust in distributed dynamic networks, such as those considered in Global Computing. The model builds on basic ideas from trust management systems and relies on domain theory to provide a semantic model for the interpretation of trust policies in trust-based security systems. Our technical contribution is based bi-ordered structures $(\mathcal{T}, \preceq, \sqsubseteq)$, where the information ordering $\sqsubseteq$ measures the information contents of data, and is needed to compute the fixpoint of mutually recursive policies, while the trust ordering $\preceq$ measures trust degrees and is used to make trust-informed decisions. Following this lead, we presented an interval construction as a canonical way to add uncertainty to trust lattices, and used the theory to guide the design and underpin the semantics of a simple, yet realistic policy language. We believe that the model can be used to explain existing trust-based systems, as well as help the design of new ones.

Of course, we are still at the first steps of development, and we still need to assess the generality of our approach is by verifying further examples. One of the main challenges ahead is to complement the denotational model presented here with an operational model where, for instance, we will need to address the question of computing trust information efficiently over the global network. Due to the great variability and absence of central control of the kind of network we are interested in, this poses serious challenges. For instance, in many applications it will not be feasible or necessary at all to compute exact values. We are therefore aiming at an operational model which can compute sufficient approximations to trust values. One issue is the update of computed trust elements; it would be interesting to investigate dynamic algorithms to update the least fixpoint. Another important issue is trust negotiation, where requester and granter engage in complex protocols aimed to convince each other of their reciprocal trustworthiness, without disclosing more evidence than necessary. Similar ideas appear in the literature as “proof carrying authentication” [1] and “automated trust negotiation” [23].

From the denotational point of view, we would like to develop a theory to account for the dynamic modification of the “web of trust.” The simplest instance of this is when a principal decides to modify its trust policy. This introduces an element of non-monotonicity that we plan to investigate using an extension of our model based on a “possible world” semantics, where updating a policy signifies the transition to a “new world” and triggers a (partial) re-computation of the global trust function.

Acknowledgements. We would like to thank Karl Krukow and the Secure project consortium and in particular the Aarhus group and the Cambridge Opera group for the development of the research. Many thanks go to Maria Vigliotti who contributed to early developments.

References

1. Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *Proc. 6th ACM Conference on Computer and Communications Security*, 1999.
2. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proc. IEEE Conference on Security and Privacy, Oakland*, 1996.

3. Matt Blaze, Joan Feigenbaum, and Jack Lacy. KeyNote: Trust management for public-key infrastructure. *LNCS*, 1550:59–63, 1999.
4. Michael Burrows, Martín Abadi, Butler W. Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *LNCS*, 576:1–23, 1991.
5. Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *In Proceedings of the Royal Society, Series A*, 426:18–36, 1991.
6. Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. REFEREE: Trust management for web applications. *Computer Networks and ISDN Systems*, 29(8-13):953–964, 1997.
7. Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. `http://theory.lcs.mit.edu/~rivest`, 1999.
8. Yun Ding, Patrick Horster, and Holger Petersen. A new approach for delegation using hierarchical delegation tokens. In *Communications and Multimedia Security*, pages 128–143, 1996.
9. Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tat Ylonen. SPKI certificate theory. *Internet RFC 2693*, 1999.
10. Tyrone Grandison and Morris Sloman. A survey of trust in internet application. *IEEE Communications Surveys, Fourth Quarter*, 2000.
11. George Gräzer. *Lattice Theory: First Concepts and Distributive Lattices*. Freeman and Company, 1971.
12. Sushil Jajodia, Pierangela Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *Proc. of the 1997 IEEE Symposium on Security and Privacy, Oakland, CA*, 1997.
13. Andrew J. I. Jones and Babak S. Firozabadi. On the characterisation of a trusting agent. In *Workshop on Deception, Trust and Fraud in Agent Societies*, 2000.
14. Adun Jøsang. A logic for uncertain probabilities. *Fuzziness and Knowledge-Based Systems*, 9(3), 2001.
15. Ulrich W. Kulish and Willard L. Miranker. *Computer Arithmetic in Theory and Practice*. Academic Press, 1981.
16. D. Harrison McKnight and Norman L. Chervany. The meanings of trust. *Trust in Cyber-Societies - LNAI*, 2246:27–54, 2001.
17. Ingmar Pörn. Some basic concepts of action. In *S. Stenlund (ed.), Logical Theory and Semantic Analysis. Reidel, Dordrecht*, 1974.
18. P. Venkat Rangan. An axiomatic basis of trust in distributed systems. In *Symposium on Security and Privacy*, 1998.
19. Ronald L. Rivest and Butler Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.
20. Dana S. Scott. Domains for denotational semantics. *ICALP '82 - LNCS*, 140, 1982.
21. Stephen Weeks. Understanding trust management systems. In *Proc. IEEE Symposium on Security and Privacy, Oakland*, 2001.
22. Uwe G. Wilhelm, Levente Buttyàn, and Sebastian Staamann. On the problem of trust in mobile agent systems. In *Symposium on Network and Distributed System Security*. Internet Society, 1998.
23. William H. Winsborough and Ninghui Li. Towards practical automated trust negotiation. In *IEEE 3rd Intl. Workshop on Policies for Distributed Systems and Networks*, 2002.
24. Philip Zimmermann. *PGP Source Code and Internals*. The MIT Press, 1995.

Recent BRICS Report Series Publications

RS-03-4 **Marco Carbone, Mogens Nielsen, and Vladimiro Sassone.** ***A Formal Model for Trust in Dynamic Networks*****. January 2003. 18 pp.**

RS-03-3 **Claude Crépeau, Paul Dumais, Dominic Mayers, and Louis Salvail.** ***On the Computational Collapse of Quantum Information*****. January 2003. 31 pp.**

RS-03-2 **Olivier Danvy and Pablo E. Martínez López.** ***Tagging, Encoding, and Jones Optimality*****. January 2003. Appears in Degano, editor,** ***Programming Languages and Systems: Twelfth European Symposium on Programming*****, ESOP '03 Proceedings, LNCS 2618, 2003, pages 335–347.**

RS-03-1 **Vladimiro Sassone and Paweł Sobocinski.** ***Deriving Bisimulation Congruences: 2-Categories vs. Precategories*****. January 2003. 28 pp. Appears in Gordon, editor,** ***Foundations of Software Science and Computation Structures*****, FoSSaCS '03 Proceedings, LNCS 2620, 2003, pages 409–424.**

RS-02-53 **Olivier Danvy.** ***A Lambda-Revelation of the SECD Machine*****. December 2003.**

RS-02-52 **Olivier Danvy.** ***A New One-Pass Transformation into Monadic Normal Form*****. December 2002. 16 pp. Appears in Hedin, editor,** ***Compiler Construction, 12th International Conference*****, CC '03 Proceedings, LNCS 2622, 2003, pages 77–89.**

RS-02-51 **Gerth Stølting Brodal, Rolf Fagerberg, Anna Östlin, Christian N. S. Pedersen, and S. Srinivasa Rao.** ***Computing Refined Buneman Trees in Cubic Time*****. December 2002. 14 pp.**

RS-02-49 **Mikkel Nygaard and Glynn Winskel.** ***HOPLA—A Higher-Order Process Language*****. December 2002. 18 pp. Appears in Brim, Jančar, Křetínský and Antonín, editors,** ***Concurrency Theory: 13th International Conference*****, CONCUR '02 Proceedings, LNCS 2421, 2002, pages 434–448.**

RS-02-48 **Mikkel Nygaard and Glynn Winskel.** ***Linearity in Process Languages*****. December 2002. 27 pp. Appears in Plotkin, editor,** ***Seventeenth Annual IEEE Symposium on Logic in Computer Science*****, Lics '02 Proceedings, 2002, pages 433–446.**