How to Exploit Ontologies in Trust Negotiation

⋆ Travis Leithead[1], Wolfgang Nejdl[2], Daniel Olmedilla[2], Kent E. Seamons[1], Marianne Winslett[3], Ting Yu[4], and Charles C. Zhang[3,5]

[1] Department of Computer Science, Brigham Young University, USA
{tleithea,seamons}@cs.byu.edu
[2] L3S Research Center and University of Hannover, Germany
{nejdl,olmedilla}@l3s.de
[3] Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA
{winslett,cczhang}@cs.uiuc.edu
[4] Dept. of Computer Science, North Carolina State University, USA
yu@csc.ncsu.edu
[5] Cisco Systems, Inc., USA
cczhang@cisco.com

Abstract. The World Wide Web makes it easy to share information and resources, but offers few ways to limit the manner in which these resources are shared. The specification and automated enforcement of security-related policies offer promise as a way of providing controlled sharing, but few tools are available to assist in policy specification and management, especially in an open system such as the Web, where resource providers and users are often strangers to one another and exact and correct specification of policies will be crucial. In this paper, we propose the use of ontologies to simplify the tasks of policy specification and administration, and to avoid several information leakage problems in run-time trust management in open systems.

1 Introduction

Open distributed environments like the World Wide Web offer easy sharing of information, but offer few options for the protection of sensitive information and other sensitive resources, such as Web Services. Proposed approaches to controlling access to Web resources include XACML [3], SAML [4], WS-Trust [2] and Liberty-Alliance[1]. All of these approaches to trust management rely on the use of vocabularies that are shared among all the parties involved, and declarative policies that describe who is allowed to do what. Some of these approaches also recognize that trust on the Web, and in any other system where resources are shared across organizational boundaries, must be *bilateral*.

Specifically, the Semantic Web provides an environment where parties may make connections and interact without being previously known to each other. In many cases, before any meaningful interaction starts, a certain level of trust must be established from scratch. Generally, trust is established through exchange of information between the two parties. Since neither party is known to the other, this trust establishment process should

⋆ In alphabetical order

be bi-directional: both parties may have sensitive information that they are reluctant to disclose until the other party has proved to be trustworthy at a certain level. As there are more service providers emerging on the Web every day, and people are performing more sensitive transactions (for example, financial and health services) via the Internet, this need for building mutual trust will become more common.

To make controlled sharing of resources easy in such an environment, parties will need software that automates the process of iteratively establishing bilateral trust based on the parties' access control policies, i.e., *trust negotiation* software. Trust negotiation differs from traditional identity-based access control and information release systems mainly in the following aspects:

1. Trust between two strangers is established based on parties' properties, which are proven through disclosure of digital credentials.
2. Every party can define access control and release policies (*policies*, for short) to control outsiders' access to their sensitive resources. These resources can include services accessible over the Internet, documents and other data, roles in role-based access control systems, credentials, policies, and capabilities in capability-based systems. The policies describe what properties a party must demonstrate (e.g., ownership of a driver's license issued by the State of Illinois) in order to gain access to a resource.
3. Two parties establish trust directly without involving trusted third parties, other than credential issuers. Since both parties have policies, trust negotiation is appropriate for deployment in a peer-to-peer architecture such as the Semantic Web, where a client and server are treated equally. Instead of a one-shot authorization and authentication, trust is established incrementally through a sequence of bilateral credential disclosures.

A trust negotiation process is triggered when one party requests to access a resource owned by another party. The goal of a trust negotiation is to find a sequence of credentials $(C_1, \ldots, C_k, R)$, where R is the resource to which access was originally requested, such that when credential C_i is disclosed, its policy has been satisfied by credentials disclosed earlier in the sequence or to determine that no such credential disclosure sequence exists.

The use of declarative policies and the automation of the process of satisfying them in the context of such a *trust negotiation process* seem to be the most promising approach to providing controlled access to resources on the Web. However, this approach opens up new and pressing questions:

1. What confidence can we have that our policies are correct? Because the policies will be enforced automatically, errors in their specification or implementation will allow outsiders to gain inappropriate access to our resources, possibly inflicting huge and costly damages. Unfortunately, real-world policies [9] tend to be as complex as any piece of software when written down in detail; getting a policy right is going to be as hard as getting a piece of software correct, and maintaining a large number of them is difficult as well.
2. How do we avoid information leakage problems in automated trust negotiation? Using very specific policies we may already leak information about what we want

to protect. On the other hand malicious opponents may try to get information which is not relevant to the resource we want to access.

In this paper, we will address these questions by exploring, in section 2, the use of ontologies for providing abstraction and structure for policy specification, and, in section 3, for providing additional knowledge which can be used during the trust negotiation process to avoid giving out to much information during the trust negotiation process.

2 Using Ontologies to Ease Policy Specification and Management

Using structure and abstraction helps for maintaining complex software and it also helps for maintaining complex sets of policies. In the context of the Semantic Web, ontologies provide formal specification of concepts and their interrelationships, and play an essential role in complex web service environments [6], semantics-based search engines [11] and digital libraries [19].

One important purpose of these formal specifications is sharing of knowledge between independent entities. In the context of trust negotiation, we want to share information about credentials and their attributes, needed for establishing trust between negotiating parties. Figure 1 shows a simple example ontology for credential IDs.

Each credential class can contain its own attributes; e.g., a Cisco Employee ID credential has three attributes: `name`, `rank` and `department`. Trust Negotiation is attributed-based and builds on the assumption that each of these attributes can be protected and disclosed separately. While in some approaches (e.g. with X.509 certificates) credentials and their attributes are signed together as a whole by the credential issuer, in this paper we will rely on cryptographic techniques such as [17] which allow us to disclose credentials with different granularities, hiding attributes not relevant to a given policy.

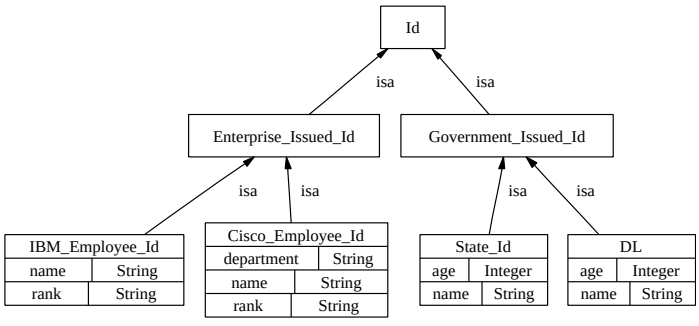

Fig. 1. Simple ID Credential Ontology

In trust negotiation, a party's security policies consist of constraints that the other party has to satisfy; e.g. it has to produce a proof that it owns a certain credential, and

that one of the credential attributes has to be within a certain range. Assuming a casino requires any customer's age to be over 21 and requires a state Id to testify that, the policy for its `admits` service can be represented as[6]:

Casino:
allowedInCasino(Requester) ←
 type(CredentialIdentifier, "State_Id") @ Issuer @ Requester,
 issuedFor(CredentialIdentifier, Requester) @ Issuer @ Requester,
 age(CredentialIdentifier, Age) @ Issuer @ Requester,
 Age $>$ 21.

In this example, the first two statements in the body of the rule require the requester to prove that he owns a credential of type `State_Id` issued by `Issuer`. If the requester proves that he has it (notice that information about attributes has not been disclosed so far, except for the `issuedFor` attribute), the casino asks for the value of the attribute `age` in the presented credential. Then it verifies whether the requester's age is over 21 and, if successful, admits the `Requester` into the casino.

2.1 Sharing Policies for Common Attributes

Often, credentials share common attributes, and these attributes might share the same policies. Figure 1 shows an example of a simple credential hierarchy, where the concrete credential classes used are depicted in the leaves of the hierarchy. The upper part of the hierarchy represent the different abstract classes: the root represents any ID, which is partitioned into different subclasses according to the issuer of the credential, distinguished between `Government_Issued` and `Enterprise_Issued` IDs. The leaf nodes represent concrete classes which contain the attributes, i.e. `name`, `age`, `rank` and `department`.

This somewhat degenerated hierarchy however does not yet allow for policy re-use. For this we have to exploit attribute inheritance. In our example, all leaf nodes share the `Name` attribute, which therefore can be moved up to the root class `Id`. We are now able to specify common policies for the `Name` attribute at the `Id` level. Similarly, we will move `Rank` up so that it becomes an attribute of `Enterprise_Issued_Id`, and `Age` an attribute of `Government_Issued_Id`. A subclass automatically inherits its superclass's attributes, which might be local or inherited from the superclass's superclass. This leads to the refined ontology as described in figure 2, where each leaf node has the same set of attributes as in figure 1, but inherited from higher levels. This makes it possible to specify shared policies for these shared attributes, similar to method inheritance in object oriented programming languages.

2.2 Composing and Overriding Policies

Now, given such credential ontologies, we can specify security policies at different levels. Being able to inherit and compose these security policies simplifies policy maintenance, though of course we have to distinguish between the case where we compose

[6] Our examples in this paper are specified using a simplified version of the PeerTrust [16, 14] language

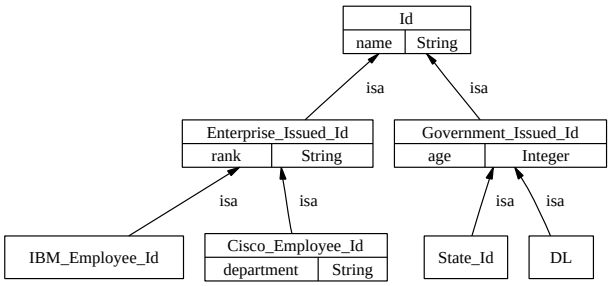

Fig. 2. Refined Ontology

inherited and local policies and the case where the policy specified for an attribute of a specific class overrides the policy inherited from the superclass. In this paper we will describe *mandatory policies* and *default polices*.

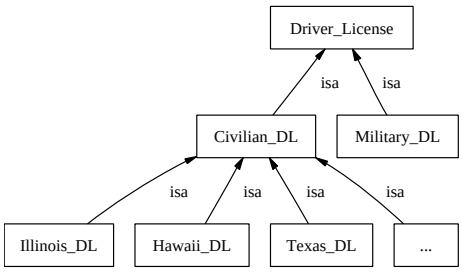

Fig. 3. Driver License Ontology

Mandatory Policies Mandatory policies are used when we want to mandate that policies of a higher level are always enforced at lower levels. Assume the ontology depicted in 3 and that we want to hire an experienced driver to accomplish a certain highly classified and challenging task. Before we show the details of the task to an interested candidate, we want the candidate to present a driver's license, which can be proven to satisfy the following mandatory policies as specified at the different levels:

At the `Driver_License` level, we enforce generic requirements for driver licenses; e.g., a driver license has to be signed by a federally authorized certificate authority and must not have expired.

At the `Civilian_DL` level, we require that the driver license is non-commercial, assuming commercial drivers may have a conflict of interests in the intended task.

At the `Illinois_DL` level, we require that the category of the driver license is not F, assuming F licenses are for farm vehicles only. At the `Military_DL` level, we

can specify policies such as “the driver license must be for land passenger vehicles” as opposed to fighter planes or submarines.

So for an Illinois driver, the overall policy is: must hold a valid driver license, as qualified by the policy at the Driver_License level; must hold a non-commercial driver license, as required by the Civilian_DL policy; and the driver license must not be for farm vehicles only. The advantage of using mandatory policies here is twofold: firstly, shared policies such as the generic driver license requirements are only specified once at a higher level, which means a more compact set of policies; secondly, it gives a cleaner and more intuitive logical structure to policies, which makes the policies easier to specify and manage.

Default Policies In this example, we assume that all driver licenses show driving experience (expressed in years of driving). Now suppose that a specific task requires the following policy: in most cases, 4 years’ driving experience is required; however, if the driver comes from Texas, he/she needs only 3 years’ experience (assuming it is harder to get a driver’s license in Texas).

To simplify the specification of this policy, we can use the default policy construct. A parent’s default policy is inherited and enforced by a child if and only if the child does not have a corresponding (overriding) policy. In our example, we can specify at the `Driver_License` level that the driving age has to be at least 4 years; then at the `Texas_DL` level, specify an overriding policy that the driving age has to be at least 3 years.

It’s of interest to note that the same result can be achieved here without using default policies: we can move the shared 4-year mandatory policy down to *every* concrete driver license class except `Texas_DL`, where we require 3 years. However, the power of policy sharing is lost.

3 Using Ontologies to Protect Sensitive Information

In the previous section we have used ontologies to structure credentials and policies, making policy maintenance easier. This section concentrates on how to use ontologies to offer additional protection for sensitive information.

3.1 Avoiding Information Leaking Requests

We assume the ontology presented in figure 2 and an equipment provider which has the following external policy for its big customer Cisco:

Cisco Purchase Records:
permits(Requester) $ Requester ←
  type(CredentialIdentifier, “Cisco_Employee_Id”) @ Issuer @ Requester,
  issuedFor(CredentialIdentifier, Requester) @ Issuer @ Requester,
  authorizedEmployee(CredentialIdentifier) @ Issuer @ Requester.
authorizedEmployee(CredentialIdentifier) ←
  rank(CredentialIdentifier, Rank),

Rank > “Manager”.
authorizedEmployee(CredentialIdentifier) ←
department(CredentialIdentifier, “Sales”).

This policy gives access to Cisco employees which are either working at the sales department or are at least a manager. If the request for a valid Cisco employee ID is already considered leakage of confidential business information, we can obscure the policy by abstracting it to a higher level in the type hierarchy:

Cisco Purchase Records:
permits(Requester) $ Requester ←
type(CredentialIdentifier, “Enterprise_Issued_Id”) @ Issuer @ Requester,
issuedFor(CredentialIdentifier, Requester) @ Issuer @ Requester,
type(CredentialIdentifier, “Cisco_Employee_Id”),
authorizedEmployee(CredentialIdentifier).

In general, we can summarize this abstraction process as follows: elevate the type of a required sensitive credential to one of its ancestors, which is more generic and discloses less information when requested from the other party; the policy body stays unchanged except that an additional type check is necessary to offset the abstraction.

3.2 Avoiding Answering Unnecessary Requests

We have focused on protecting sensitive information on behalf of the requester of a resource. Similarly, the party providing resources also wants to disclose only information that is relevant to the task at hand. Ontologies describing standard types of negotiations help accomplish this goal. These ontologies contain properties that will describe typical attributes required in the specified negotiation, without specifying any additional constraints. A simple ontology of this kind is depicted in figure 4. This kind of ontology leads to two additional uses of ontologies: *need-to-know disclosure* and *predisposed negotiation*.

Need-to-Know Disclosure. A negotiator may use properties collected from such ontologies to screen received policies and detect unusual credential solicitations, a technique we call need-to-know disclosure. Need-to-know disclosure occurs when the service provider possesses the credentials and properties necessary to solicit a requester’s sensitive property, but the property in question is not relevant to the current negotiation. For example, a trust negotiation agent is entering into a home loan transaction with a legitimate bank. The bank releases a policy requiring a credit card. The requester’s trust agent ascertains that the credit card request is not relevant to the home loan negotiation because that property is not found within the ontology for the current negotiation. Though the bank server - acting as a legitimate financial institution - could request the credit card, the home loan transaction as defined by the given ontology, doesn’t typically require a credit card. Thus, information that the bank server did not need-to-know is not disclosed. Need-to-know disclosure based on knowledge assimilated with the aid of ontologies facilitates a safeguard to the resource requester against malicious property phishing attacks or poorly formed policies without revealing the presence or absence of the requested property.

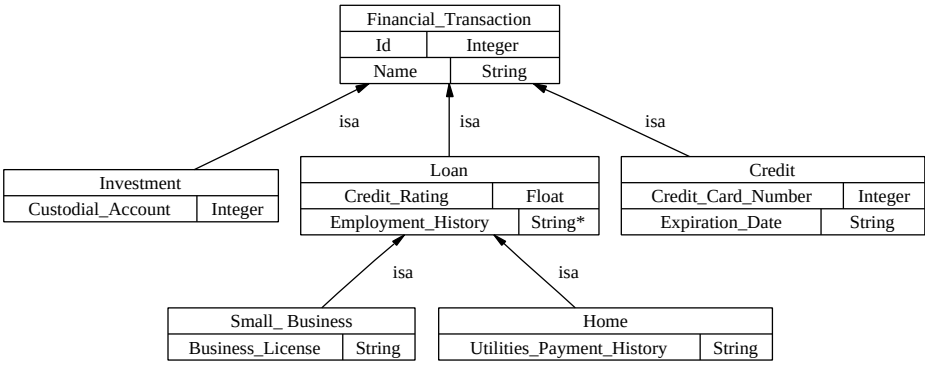

Fig. 4. Negotiation-Type Ontology with Associated Properties for a Typical Financial Transaction

Predisposed Disclosure Strategies. Sometimes a party does not want to wait for specific detailed requests for certificates, but provides information about a set of predisposed credential or policies in one step. This is a modification of the eager strategy described in [21]. Methods similar to those described in need-to-know disclosure provide the basic framework - a set of properties related to a negotiation type which are collected from an ontology (we leave the details of the discovery and collection of these ontologies for future work). Predisposed disclosure involves selecting the credentials or policies that best fit the anticipated needs of the service provider and pushing them along with the resource request. Predisposed disclosure expedites rounds of negotiation in an attempt to satisfy the service provider's disclosure policy without receiving it. This disclosure technique compliments the eager strategy by narrowing the scope of the credentials disclosed to a more relevant set (preserving the sensitivity of credentials that do not pertain to the negotiation) and ensuring the privacy of sensitive policies, since the eager strategy requires no policy disclosures.

4 Related Work

Recent work in the context of the Semantic Web has focused on how to describe security requirements. KAoS and Rei policy languages [15, 20] investigate the use of ontologies for modeling speech acts, objects, and access types necessary for specifying security policies on the Semantic Web. Hierarchies of annotations to describe capabilities and requirements of providers and requesting agents in the context of Web Services are introduced in [10]. Those annotations are used during the matchamaking process to decide if requester and provider share similar security characteristics and if they are compatible. Ontologies have also been discussed in the context of digital libraries for concepts and credentials [5]. An approach called "most specific authorization" is used for conflict resolution. It states that policies specified on specific elements prevail over policies specified on more general ones. In this paper we explore complementary uses of ontologies for trust negotiation, through which we target iterative trust establishment between strangers and the dynamic exchange of credentials during an iterative trust negotiation

process that can be declaratively expressed and implemented. Work done in [8] defines *abstractions* of credentials and services. Those abstractions allow a service provider to request for example a credit card without specifically asking for each kind of credit card that it accepts. We add to this work in the context of policy specification the concept of *mandatory* and *default* policies.

Ontology-based policy composition and conflict resolving have also been been discussed in previous work. Policy inheritance is done by *implication* in [12], but it does not provide any fine-grained overriding mechanism based on class levels. *Default properties* are discussed in [13], short of generalizing the idea to policies. The approaches closest to our default and mandatory policy constructs are the *weak* and *strong* authorizations in [7], where a strong rules always overrides a weak rule, and SPL in [18], which forces the security administrator to combine policies into a structure that precludes conflicts. Compared to these approaches, we find ours particularly simple and intuitive, while its expressiveness well serves general trust negotiation needs.

The concepts forming the basis for need-to-know credential disclosures within automated trust negotiation are suggested in [21], where two dichotomous negotiation strategies are meticulously analyzed: eager and parsimonious. The strength of the eager strategy is its simplicity, yet it discloses all non-sensitive credentials, which raises a privacy concern. A need for a smarter, need-to-know, credential disclosure strategy is recommended in which credentials relevant only to the present negotiation are disclosed. The parsimonious strategy focuses and guides a negotiation, but does so only according to the received credential request policies. Our work relies upon ontologies to provide additional semantics that supplement these negotiation strategies and enable genuine need-to-know disclosure.

5 Conclusions and Future Research Directions

Ontologies can provide important supplemental information to trust negotiation agents both at compile time (to simplify policy management and composition) and at run-time (to avoid certain forms of information leakage for all peers participating in a negotiation). This paper has explored some important benefits of using ontologies.

For compile time usage, ontologies with their possibility of sharing policies for common attributes provide an important way for structuring available policies. In this context we discussed two useful strategies to compose and override these policies, building upon the notions of mandatory and default policies. Further investigations into these mechanisms should draw upon previous work on inheritance reasoning and method composition in object oriented programming languages, and will improve the maintenance of large sets of policies for real-world applications.

For run-time usage, ontologies provide valuable additional information, which can be used to avoid information leaking requests and enable negotiators to detect credential requests irrelevant for the service currently negotiated. Need-to-know disclosure, and predisposed negotiation are two ways to harness ontologies in a useful manner at runtime to provide such privacy benefits.

Need-to-know disclosures make assumptions that require further research: the existence of standard negotiation typing ontologies; implicit trust of the content of such

ontologies and ontologies in general; determination of credential relevancy and efficiency models for trust negotiation which include the overhead of using ontologies. For predisposed negotiation, further work is required to determine how local credential sets are condensed to reduce the scope of a negotiation and to select appropriate credentials to push. Analysis of the effects of predisposed negotiation should focus on the effects of creating a localized set of negotiation parameters (policies and credentials) specific to the scope of the current negotiation, as well as the overall effects of ontology information inference.

Acknowledgments

The research of Nejdl and Olmedilla was partially supported by the projects ELENA (http://www.elena-project.org, IST-2001-37264) and REWERSE (http://rewerse.net, IST-506779). The research of Leithead, Seamons and Winslett was supported by DARPA (N66001-01-1-8908), the National Science Foundation (CCR-0325951,IIS-0331707) and The Regents of the University of California. The research of Yu was partially supported by Faculty Research and Professional Development Fund, NCSU.

References

1. *Liberty Alliance Project. http://www.projectliberty.org/about/whitepapers.php*.
2. *Web Services Trust Language (WS-Trust) Specification. http://www-106.ibm.com/developerworks/library/specification/ws-trust/*.
3. Xacml 1.0 specification http://xml.coverpages.org/ni2003-02-11-a.html.
4. Assertions and protocol for the oasis security assertion markup language (saml); committee specification 01, 2002.
5. N. R. Adam, V. Atluri, E. Bertino, and E. Ferrari. A content-based authorization model for digital libraries. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):296–315, 2002.
6. A. Ankolekar. Daml-s: Semantic markup for web services.
7. E. Bertino, S. Jojodia, and P. Samarati. Supporting multiple access control policies in database systems. In *IEEE Symposium on Security and Privacy*, pages 94–109, Oakland, CA, 1996. IEEE Computer Society Press.
8. P. Bonatti and P. Samarati. Regulating Service Access and Information Release on the Web. In *Conference on Computer and Communications Security*, Athens, Nov. 2000.
9. Cassandra policy for national ehr in england. http://www.cl.cam.ac.uk/users/mywyb2/publications/ehrpolicy.pdf.
10. G. Denker, L. Kagal, T. Finin, M. Paolucci, and K. Sycara. Security for daml web services: Annotation and matchmaking. In *Proceedings of the 2nd International Semantic Web Conference*, Sanibel Island, Florida, USA, Oct. 2003.
11. M. Erdmann and R. Studer. How to structure and access xml documents with ontologies. *Data and Knowledge Engineering*, 36(3), 2001.
12. W. Emayr, F. Kastner, G. Pernul, S. Preishuber, and A. Tjoa. Authorization and access control in iro-db.
13. R. Fikes, D. McGuinness, J. Rice, G. Frank, Y. Sun, and Z. Qing. Distributed repositories of highly expressive reusable knowledge, 1999.

14. R. Gavriloaie, W. Nejdl, D. Olmedilla, K. Seamons, and M. Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *1st First European Semantic Web Symposium*, Heraklion, Greece, May 2004.
15. L. Kagal, T. Finin, and A. Joshi. A policy based approach to security for the semantic web. In *2nd International Semantic Web Conference*, Sanibel Island, Florida, USA, Oct. 2003.
16. W. Nejdl, D. Olmedilla, and M. Winslett. PeerTrust: automated trust negotiation for peers on the semantic web. In *Workshop on Secure Data Management in a Connected World (SDM'04)*, Toronto, Aug. 2004.
17. P. Persiano and I. Visconti. User privacy issues regarding certificates and the tls protocol. In *Conference on Computer and Communications Security*, Athens, Nov. 2000.
18. C. Ribeiro and P. Guedes. Spl: An access control language for security policies with complex constraints, 1999.
19. S. B. Shum, E. Motta, and J. Domingue. Scholonto: an ontology-based digital library server for research documents and discourse. *Int. J. on Digital Libraries*, 3(3):237–248, 2000.
20. G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok. Semantic web languages for policy representation and reasoning: A comparison of KAoS, Rei and Ponder. In *2nd International Semantic Web Conference*, Sanibel Island, Florida, USA, Oct. 2003.
21. W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. DARPA Information Survivability Conference and Exposition, IEEE Press, Jan 2000.

Semantic Web Publishing using Named Graphs

Jeremy J. Carroll[1], Christian Bizer[2], Patrick Hayes[3], and Patrick Stickler[4]

[1] Hewlett-Packard Labs,Bristol, UK
[2] Freie Universität Berlin, Germany
[3] IHMC, Florida, USA
[4] Nokia, Tampere, Finland

Abstract. The Semantic Web consists of many RDF graphs nameable by URIs. This paper extends the syntax and semantics of RDF to cover such Named Graphs. This enables RDF statements that describe graphs, which is beneficial in many Semantic Web application areas. In this paper, we explore the application area of Semantic Web publishing: Named Graphs allow publishers to communicate assertional intent, and to sign their graphs; information consumers can evaluate specific graphs using task-specific trust policies, and act on information from those Named Graphs that they accept. Graphs are trusted depending on: their content; information about the graph; and the task the user is performing. The extension of RDF to Named Graphs provides a formally defined framework to be a foundation for the Semantic Web trust layer.

1 Introduction

A simplified view of the Semantic Web is a collection of web retrievable RDF documents, each containing an RDF graph. The RDF Recommendation [4, 11, 19, 23], explains the meaning of any one graph, and how to merge a set of graphs into one, but does not provide suitable mechanisms for talking about graphs or relations between graphs. The ability to express metainformation about graphs is required for:

Data syndication systems need to keep track of provenance information, and provenance chains.

Restricting information usage Information providers might want to attach information about intellectual property rights or their privacy preferences to graphs in order to restrict the usage of published information [15, 25].

Access control A triple store may wish to allow fine-grain access control, which appears as metadata concerning the graphs in the store [21].

Signing RDF graphs As discussed in [12], it is necessary to keep the graph that has been signed distinct from the signature, and other metadata concerning the signing, which may be kept in a second graph.

Expressing propositional attitudes such as modalities and beliefs [20].

RDF reification has well-known problems in addressing these use cases as previously discussed in [14]. To avoid these problems several authors propose quads [3, 16, 21, 24], consisting of an RDF triple and a further URIref or blank node or ID. The proposals vary widely in the semantic of the fourth element, using it to refer to information sources, to model IDs or statement IDs or more generally to 'contexts'.

We propose a general and simple variation on RDF, using sets of *named* RDF graphs. A set of Named Graphs is a collection of RDF graphs, each one of which is named with a URIref. The name of a graph may occur either in the graph itself, in other graphs, or not at all. Graphs may share URIrefs but not blank nodes.

Named Graphs can be seen as a reformulation of quads in which the fourth element's distinct syntactic and semantic properties are clearly distinguished, and the relationship to RDF's triples, abstract syntax and semantics is clearer.

We describe how Named Graphs can be used for Semantic Web publishing, looking in particular on provenance tracking and how it interacts with the choices made by consumers of Semantic Web information about which information to trust.

2 Abstract Syntax and Semantics of Named Graphs

RDF syntax is based on a mathematical abstraction: an RDF graph is defined as a set of triples. These graphs are represented by documents which can be retrieved from URIs on the Web. Often these URIs are also used as a name for the graph, for example with an `owl:imports`. To avoid confusion between these two usages we distinguish between Named Graphs and the RDF graph that the Named Graph encodes or represents. A Named Graph is an entity with two functions *name* and *rdfgraph* defined on it which determine respectively its name, which is a URI , and the RDF graph that it encodes or represents. These functions assign a unique name and RDF graph to each Named Graph, but Named Graphs may have other properties; and named graphs may be concrete resources rather than set-theoretic abstractions. We follow the RDF convention whereby graphs which are equivalent in the sense of [23] - i.e. which differ only in the identity of their blank nodes - are considered to be identical. This has the consequence that blank nodes are considered to be internal to a graph, i.e. that two distinct RDF graphs do not have any blank nodes in common.

In more detail, we define a set of Named Graphs $\mathbf{N}$ to be a function from a set of URI references to a set of RDF graphs, i.e. a set of pairs $\langle N, G \rangle$ where G is an RDF graph.[5] Each pair $ng = (n, g) \in N$ is a Named Graph in $\mathbf{N}$, and we write $n = \mathit{name}(ng)$ and $g = \mathit{rdfgraph}(ng)$.

An RDF interpretation I (as in [19]) *conforms* with a set of Named Graphs $\mathbf{N}$ when:

For every Named Graph $ng \in \mathbf{N}$, we have $I(\mathit{name}(ng)) = ng$

Note that the Named Graph itself, rather than the RDF graph it intuitively "names", is the denotation of the name. We consider the RDF graph to be related to the Named Graph in a way analogous to that in which a class extension is related to a class in RDFS. This 'intensional' (c.f. [19]) style of modelling allows for distinctions between several 'copies' of a single RDF graph and avoids pitfalls arising from accidental identification of similar Named Graphs. Note also that in any conforming interpretation, named graphs are denoted by their labelling URIs and hence are first-class elements of the universe of discourse on exactly the same basis as all other resources.

[5] We have removed the legacy constraint that a literal cannot be the subject of a triple.

As noted, we follow the notion of graph equivalence defined in RDF [23] by treating two RDF graphs which differ only in the identity of their blank nodes as being the same graph. A more explicit approach would take graph equivalence from [23] (i.e. a 1:1 mapping on blank nodes, a *renaming* function), and say that a *nameblanked* RDF graph is an equivalence class under this equivalence relation of replacing blank nodes by other blank nodes under some renaming. Then the *rdfgraph* of a Named Graph is a *nameblanked* RDF graph. We generally will ignore this complication.

2.1 RDF Reification

A 'reified statement' [19] is a single RDF statement described and identified by a URIreference. Within the framework of this paper, it is natural to think of this as a Named Graph containing a single triple, blurring the distinction between a (semantic) statement and a (syntactic) triple. This provides a useful connection with the traditional use of reification and a potential migration path.

2.2 Accepting Graphs

A set of Named Graphs $\mathbf{N}$ is not given a single formal meaning. Instead, the formal meaning depends on an additional set $A \subset \text{domain}(N)$. A identifies some of the graphs in the set as *accepted*. Thus there are $2^{|\text{domain}(N)|}$ different formal meanings associated with a set of Named Graphs, depending on the choice of A. The meaning of a set of accepted Named Graphs $\langle A, \mathbf{N} \rangle$ is given by taking the graph merge $\bigcup_{a \in A} N(a)$, and then interpreting that graph with the RDF semantics [19] (or an extension), subject to the additional constraint that all interpretations I conform with $\mathbf{N}$.

The choice of A reflects that the individual graphs in the set may have been provided by different people, and that the information consumers who use the Named Graphs make different choices as to which graphs to believe. Thus we do not provide one correct way to determine the 'correct' choice of A, but provide a vocabulary for information providers to express their intensions, and suggest techniques with which information consumers might come to their own choice of which graphs to accept.

3 Concrete Syntaxes and Query Languages

A concrete syntax for Named Graphs has to exhibit the name, the graph and the association between them. We offer three concrete syntaxes: TriX and RDF/XML both based on XML; and TriG as a compact plain text format.

The TriX[14] serialization is an XML format which corresponds fairly directly with the abstract syntax, allowing the effective use of generic XML tools such as XSLT, XQuery, while providing syntax extensibility using XSLT. TriX is defined with a short DTD, and also an XML Schema.

In this paper we use TriG as a compact and readable alternative to TriX. TriG is a variation of Turtle [5] which extends that notation by using '{' and '}' to group triples into multiple graphs, and to precede each by the name of that graph. The following TriG document contains two graphs. The first graph contains information about itself. The second graph refers to the first one, (namespace prefix definitions omitted).

```
:G1 { _:Monica ex:name "Monica Murphy" .
      _:Monica ex:email <mailto:monica@murphy.org> .
      :G1 pr:disallowedUsage pr:Marketing }

:G2 { :G1 ex:author :Chris .
      :G1 ex:date "2003-09-03"^^xsd:date }
```

Named Graphs are backward compatible with RDF. A collection of RDF/XML [4] documents on the Web map naturally into the abstract syntax, by using the first xml:base declaration in the document or the URL from which an RDF/XML file is retrieved as a name for the graph given by the RDF/XML file.

There are currently two query languages for Named Graphs: RDFQ [27] uses an RDF vocabulary to structure queries. TriQL [7] is a graph patterns based query language inspired by RDQL [26]. A prototypical implementation of TriX, TriG and TriQL is described in [9].

4 Semantic Web Publishing

One application area for Named Graphs is publishing information on the Semantic Web. This scenario implies two basic roles embodied by humans or their agents: Information providers and information consumers. Information providers publish information together with meta-information about its intended assertional status. Additionally, they might publish background information about themselves, e.g. their role in the application area. They may also decide to digitally sign the published information. Information providers have different levels of knowledge, and different intentions and different views of the world. Thus seen from the perspective of an information consumer, published graphs are claims by the information providers, rather than facts.

Different tasks require different levels of trust. Thus information consumers will use different trust policies to decide which graphs should be accepted and used within the specific application. These trust policies depend on the application area, the subjective preferences and past experiences of the information consumer and the trust relevant information available. A naive information consumer might for example decide to trust all graphs which have been explicitly asserted. This trust policy will achieve a high recall rate but is easily undermineable by information providers publishing false information. A more cautious consumer might require graphs to be signed and the signers to be known through a Web-of-Trust mechanism. This policy is harder to undermine, but also likely to exclude relevant information, published by unknown information providers.

4.1 Authorities, Authorization and Warrants

Information providers using RDF do not have any explicit way to express any intention concerning the truth-value of the information described in a graph; RDF does not provide for the expression of *propositional attitudes*. Information consumers may require this, however. Note that this is in addition to trust policies, and may be required in order to put such policies into operation. For example a simple policy could be: believe anything asserted by a trusted source. In order to apply this, it is necessary to have a clear

record of what is *asserted* by the source. Not all information provided by a source need be asserted by that source. We propose here a vocabulary and a set of concepts designed to enable the uniform expression of such propositional attitudes using named graphs.

We take three basic ideas as primitive: that of an *authority*, a relationship of *authorizing*, and a *warrant*. An authority is a 'legal person'; that is, any legal or social entity which can perform acts and undertake obligations. Examples include adult humans, corporations and governments. The 'authorizing' relationship holds between an authority or authorities and a Named Graph, and means that the authority in some sense commits itself to the content expressed in the graph. Whether or not this relationship in fact holds may depend on many factors and may be detected in several ways (such as the Named Graph being published or digitally signed by the authority). Finally, a warrant is a resource which records a particular propositional stance or intention of an authority towards a graph. A warrant asserts (or denies or quotes) a Named Graph and is authorized by an authority. One can think of warrants as a way of reducing the multitude of possible relationships between authorities and graphs to a single one of authorization, and also as a way of separating questions of propositional attitude from issues of checking and recording authorizations. The separation of authority from intention also allows a single warrant to refer to several graphs, and for a warrant to record other properties such as publication or expiry date.

To describe the two aspects of a warrant we require vocabulary items: a property `swp:authority` (where `swp:` is a namespace for Semantic Web publishing) relating warrants to authorities, and another to describe the attitude of the authority to the graph being represented by the warrant. We will consider two such intentions expressed by the properties `swp:assertedBy` and `swp:quotedBy`. These take a named graph as a subject and a `swp:Warrant` as object; `swp:authority` takes a warrant as a subject and a `swp:Authority` as an object. Each warrant must have a unique authority, so `swp:authority` is an OWL functional property. Intuitively, `swp:assertedBy` means that the warrant records an endorsement or assertion that the graph is true, while `swp:quotedBy` means that the graph is being presented without any comment being made on its truth. This latter is particularly useful when republishing graphs as part of a syndication process, the original publisher may assert a news article, but the syndicator, acting as a common carrier, merely provides the graph as they found it, without making any commitment as to its truth. Warrants may also be signed, and the property `swp:signatureMethod` can be used to identify the signature technique.

4.2 Warrant Descriptions as Performatives

A warrant, as described above, is a social act. However, it is often useful to embody social acts with some record; for example a contract (which is a social act) may be embodied in a document, which is identified with that act, and is often signed. In this section, we introduce the notion of a *warrant graph*, which is a Named Graph describing a warrant, that is identified with the social act. Thus, this is a resource which is both a `swp:Warrant` and an `rdfg:Graph`. Consider a graph containing a description of a warrant of another Named Graph, such as:

```
{ :G2 swp:assertedBy _:w .
```

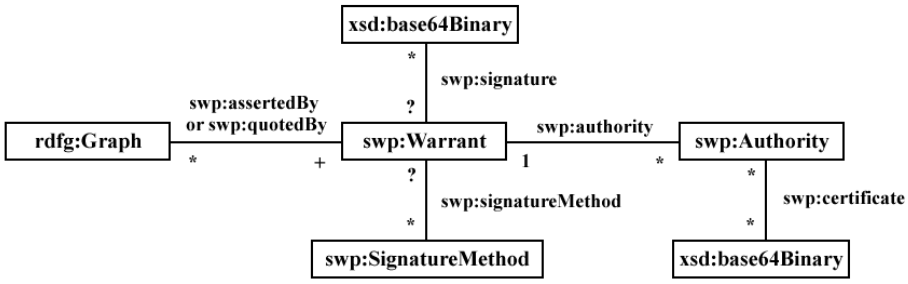

Fig. 1. The Semantic Web Publishing Vocabulary

```
  _:w rdf:type swp:Warrant[6] .
  _:w swp:authority _:a .
  _:a rdf:type swp:Authority .
  _:a foaf:mbox <mailto:chris@bizer.de> }
```

The graph is true when there is a genuine warrant; but so far we have no way to know whether this is in fact the case. A slight modification identifies the graph with the warrant itself:

```
:G1  { :G2 swp:assertedBy :G1 .
       :G1 swp:authority _:a .
       _:a foaf:mbox <mailto:chris@bizer.de> }
```

and the graph describes itself as being a warrant. Suppose further that such a *warrant graph* is in fact authorized by the authority it describes - in this case, by Chris Bizer, the owner of the mailbox: this might be established for example by being published on Chris' website, or by being digitally signed by him, or in some other way, but all that we require here is that it is in fact true. Under these circumstances, the warrant graph has the intuitive force of a first-person statement to the effect "I assert `:G2`" made by Chris.

In natural language, the utterance of such a self-describing act is called a *performative*; that is, an act which is performed by saying that one is doing it. Other examples of performatives include promising, naming and, in some cultures, marrying [2]. The key point about performatives are that while they are descriptions of themselves, they are not only descriptions: rather, the act of uttering the performative is understood to be the act that it describes. Our central proposal for how to express propositional attitudes on the Web is to treat a warrant graph as a record of a performative act, in just this way.[7] With this convention, Chris can assert the graph `:G2` by authorizing the warrant graph shown above, for by doing so he creates a warrant: the warrant graph becomes the (self-describing) warrant of the assertion of `:G2` by Chris. In order for others to

[6] The type triples are implied by domain and range constraints and can be omitted.

[7] The Bank of England uses this technique, by having each twenty pound note bear the text: "I promise to pay the bearer on demand the sum of twenty pounds."

detect and confirm the truth of this warrant requires some way to check or confirm the relationship of authorization, of course: but the qualification of the warrant graph as a warrant depends only on the relationship holding.

A graph describing a warrant is not required to be self-describing in order to be true (it may be true by virtue of some other warrant) and a warrant graph may not in fact be a performative warrant (if it is not authorized by the authority it claims). In the latter case the graph must be false, so self-describing warrant graphs whose authorization cannot be checked should be treated with caution. The warrant graph may itself be the graph asserted. Any Named Graph which has a warrant graph as a subgraph and is appropriately authorized satisfies the conditions for being a performative warrant of itself. For example:

```
:G2 { :Monica ex:name "Monica Murphy" .
      :G2 swp:assertedBy :G2 .
      :G2 swp:authority _:a .
      _:a foaf:mbox <mailto:patrick.stickler@nokia.com> . %%@
}
```

when authorized by Patrick Stickler, becomes a performative warrant for its own assertion, as well as being warranted by the earlier example. As this example indicates, a Named Graph may have a number of independent warrants.

4.3 Publishing with Signatures

Information providers may decide to digitally sign graphs, when they wish to allow information consumers to have greater confidence in the information published. For instance, if Patrick has an X.509 certificate [22], he can sign two graphs in this way:

```
:G1 { :Monica ex:name "Monica Murphy" .
      :G1  swp:assertedBy _:w1 .
      _:w1 swp:authority _:a .
      _:a  foaf:mbox <mailto:chris@bizer.de> }
:G2 { :G1 swp:quotedBy _:w2 .
      _:w2 swp:signatureMethod %%@
swp:std-method-A^^xsd:anyURI .
      _:w2 swp:signature "..."^^xsd:base64Binary .
      _:w2 swp:authority _:s .
      _:s  swp:certificate "..."^^xsd:base64Binary .
      _:s  foaf:mbox <mailto:patrick.stickler@nokia.com> .
      :G2  swp:assertedBy :G2 .
      :G2  swp:signatureMethod %%@
swp:std-method-A^^xsd:anyURI .
      :G2  swp:authority _:s .
      :G2  swp:signature "..."^^xsd:base64Binary }
```

Note that `:G2` is a warrant graph. The `swp:signature` gives a binary signature of the graph related to the warrant. Some method of forming the signature has to be agreed. This is indicated by the value of the `swp:signatureMethod` property on the warrant. We require it to be a literal URI, which can be dereferenced on the Web to retrieve a document. The document describes the method of forming the signature in detail.

Such a method could specify, for example, a variation of the graph canonicalization algorithms provided in [12][8], and choosing one of the XML canonicalization methods and one of the signature methods supported by XML Signatures [17]. Rather than make a set of decisions about these methods, we permit the warrant to indicate the methods used by including the URL of a document that contains those decisions. The URL used by the publisher needs to be understood by the information consumer, so only a few well-known variations should be used.

The publisher may choose to sign graphs to ensure that the maximum number of Semantic Web agents believe them and act on the publication. Using signatures does not modify the theoretical semantics of assertion, which is boolean; but it will modify the operational semantics, in that without signatures, any assertions made, will only be acted on by the more trusting Semantic Web information consumers, who do not need verifiable information concerning who is making them.

The formal semantics of the Semantic Web publishing vocabulary are described in more detail in [13].

4.4 The Information Consumer

The information consumer needs to decide which graphs to accept. This decision may depend on information concerning who said what, and whether it is possible to verify such information. It may also depend on the content of what has been said. We consider the use case in which an information consumer has read a set of Named Graphs off the Web. In terms of the semantics of Named Graphs (section 2.2), the information consumer needs to determine the set A. Information about the graphs may be embedded within the set of Named Graphs, hence most plausible trust policies require that we are able to provisionally understand the Named Graphs in order to determine, from their content, whether or not we wish to accept them. This is similar to reading a book, and believing it either because it says things you already believe, or because the author is someone you believe to be an authority: either of these steps require reading at least some of the book.

The trust policy an information consumer chooses for determining his set of accepted graphs depends on the application area, his subjective preferences and past experiences and the trust relevant information available. Trust policies can be based on the following types of information [10]:

First-hand information published by the actual information provider together with a graph, e.g. information about the intended assertional status of the graph or about the role of the information provider in the application domain. Example policies using the information provider's role are: "Prefer product descriptions published by the manufacturer over descriptions published by a vendor" or "Distrust everything a vendor says about its competitor."

Information published by third parties about the graph (e.g. further assertions) or about the information provider (e.g. ratings about his trustworthiness within a specific application domain). Most trust architectures proposed for the Semantic Web

[8] It is necessary to exclude the last `swp:signature` triple, from the graph before signing it: this step needs to be included in the method.

fall into this category [1, 6, 18]. These approaches assume explicit and domain-specific trust ratings. Providing such ratings and keeping them up-to-date puts an unrealistically heavy burden on information consumers.

The content of a graph together with rules, axioms and related content from graphs published by other information providers. Example policies following this approach are "Believe information which has been stated by at least 5 independent sources." or "Distrust product prices that are more than 50% below the average price."

Information created in the information gathering process like the retrieval date and the retrieval URL of a graph or the information whether a warrant attached to a graph is verifiable or not.

Example trust policies and an example algorithm for choosing which graphs to accept are found in [13]. Further example policies are found in [8, 10].

5 Conclusions

Having a clearly defined abstract syntax and formal semantics Named Graphs provide greater precision and potential interoperablity than the variety of *ad hoc* RDF extensions currently used. Combined with specific further vocabulary, this will be beneficial in a wide range of application areas and will allow the usage of a common software infrastructure spanning these areas.

The ability of self-reference combined with the Semantic Web Publishing vocabulary addresses the problem of differentiating asserted and non-asserted forms of RDF and allows information providers to express different degrees of commitment towards published information.

Linking information to authorities and optionally assuring these links with digital signatures gives information consumers the basis for using a wirde range of different task-specific trust-policies.

Further related work can be found at the TriX and Named Graphs web-site `http://www.w3.org/2004/03/trix/`.

References

1. R. Agrawal, P. Domingos, and M. Richardson. Trust Management for the Semantic Web. In *Proceedings of the 2nd ISWC*, 2003.
2. J. L. Austin. *How to do things with words*. Harvard University Press, 1962.
3. D. Beckett. Redland Notes - Contexts. `http://www.redland.opensource.ac.uk/notes/contexts.html`, 2003.
4. D. Beckett. RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2004.
5. D. Beckett. Turtle - Terse RDF Triple Language. `http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/`, 2004.
6. C. Bizer. Semantic Web Trust and Security Resource Guide. `http://www.wiwiss.fu-berlin.de/suhl/bizer/SWTSGuide`, 2004.
7. C. Bizer. TriQL - A Query Language for Named Graphs. `http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQL/`, 2004.

8. C. Bizer. TriQL.P - A Query Language for Querying Named Graphs Published by Untrustworthy Sources. http://www.wiwiss.fu-berlin.de/suhl/bizer/triqlp/, 2004.
9. C. Bizer and R. Cyganiak. NG4J - Named Graphs API for Jena. http://www.wiwiss.fu-berlin.de/suhl/bizer/NG4J/, 2004.
10. C. Bizer and R. Oldakowski. Using Context- and Content-Based Trust Policies on the Semantic Web. In *13th World Wide Web Conference, WWW2004 (Poster)*, 2004.
11. D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0. http://www.w3.org/TR/rdf-schema/, 2004.
12. J. J. Carroll. Signing RDF Graphs. In *2nd ISWC*, volume 2870 of *LNCS*. Springer, 2003.
13. J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named Graphs, Provenance and Trust, 2004. Hewlett Packard Labs, Technical Report HPL-2004-57.
14. J. J. Carroll and P. Stickler. RDF Triples in XML. In *Extreme Markup Languages 2004*, 2004.
15. Creative Commons Website. http://creativecommons.org/, 2003.
16. E. Dumbill. Tracking Provenance of RDF Data. Technical report, ISO/IEC, 2003.
17. D. Eastlake, J. Reagle, and D. Solo. XML-Signature Syntax and Processing, RFC 3275. http://www.w3.org/TR/xmldsig-core/, 2002.
18. J. Golbeck, B. Parsia, and J. Hendler. Trust Networks on the Semantic Web. In *In Proceedings of the 7th International Workshop on Cooperative Intelligent Agents, CIA2003*, 2003.
19. P. Hayes. RDF Semantics. http://www.w3.org/TR/rdf-mt/, 2004.
20. A. Ibrahim. Agent Communication Languages (ACL). http://www.engr.uconn.edu/~ibrahim/publications/acl.htm, 2000.
21. Intellidimension. RDF Gateway - Database Fundementals. http://www.intellidimension.com/pages/rdfgateway/dev-guide/db/db.rsp, 2003.
22. ITU-T. Information Technology - Open Systems Interconnection - The Directory Authentication Framework. X.509, 1997.
23. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-concepts/, 2004.
24. R. MacGregor and I.-Y. Ko. Representing Contextualized Data using Semantic Web Tools. In *Practical and Scalable Semantic Systems (workshop at 2nd ISWC)*, 2003.
25. M. Marchiori. The platform for privacy preferences. http://www.w3.org/TR/P3P/, 2002.
26. A. Seaborne. RDQL - A Query Language for RDF. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109, 2004.
27. P. Stickler. RDFQ. http://sw.nokia.com/rdfq/RDFQ.html, 2004.

The Impact of Context on the Trustworthiness of Communication: An Ontological Approach

Santtu Toivonen[1] and Grit Denker[2]

[1] VTT Information Technology
P.O.Box 1203, FIN-02044 VTT, FINLAND
santtu.toivonen@vtt.fi
[2] SRI International
Menlo Park, CA 94025, USA
grit.denker@sri.com

Abstract. We outline a Semantic Web approach for considering the impact of context information on the trustworthiness of communication. We show that the contexts of message sender, receiver, and mediating network can have influence on the degree of trust the receiver assigns to a message. We define ontologies to capture context-sensitive messaging and trust, as well as propose trust evaluation functions. We illustrate their usage with rules that determine trust factors of context information.

1 Introduction

Messages conveyed to us in everyday life often have an influence on our decisions and behaviors. The more trustworthy a message is considered, the higher will be its impact on the receiver. Trust is thereby a very significant commodity in communication. But how is trust established? A variety of factors influences trust establishment, among them integrity, reputation, credibility, reliability, congruity, predictability, and responsibility (cf. proceedings of the iTrust [12], conferences on trust management, and [11, 1]).

The decision whether or not to trust a piece of information can depend on many contextual factors including creator (who) of the data (what), time (when), location (where), and intent (why) of origination, social context of receiver and many more. Generally, context information can characterize a situation of any entity—be it a person, place, or object—that is relevant to the interaction between a user and an application, including the user and the application themselves [17]. We argue that for message-based communication, a notion of context information needs to be taken into consideration to determine the trustworthiness of a message.

A specific advantage of making context explicit in message exchanges is that this information can be used in trust policies. For example, a policy can state that news information related to a particular location is to be trusted more if the reporting entity was at the location at the time when the event occurred. In this sense, policies define how to process context information to derive trustworthiness assertions. On the basis of our integrated context and message ontology, defined in OWL [15], we will illustrate sample trust policies and discuss reasoning strategies.

We start the paper with a motivating scenario that illustrates the usefulness of integrating context information in messaging systems (see Section 2). Generalizing from the observations of the example in Section 3, we address features of context information and consider its relationship with trustworthiness to define a context-sensitive messaging ontology and a trust ontology that takes context into account. The use of context information in rules to determine its impact factor for trust is presented in Section 4. In this section we also propose evaluation function to determine a context-sensitive trust value for a message. We then present some related work in Section 5, and give a general discussion about the idea in Section 6. Finally, we close by drawing concluding remarks and future research directions in Section 7.

2 Motivating Example: An Accident Report

Consider a case where on his way to work Jim is witnessing the town's city hall caught in fire. He decides to share this information with his local friend Jane, who he knows of currently being abroad. Jim is close to the scene of the accident, along with a huge number of other curious people. Jane is on vacation on a remote island with her friends. In this example the sender's, i.e. Jim's, relevant context attributes are his location, his activity, and the time of sending the message. The network context concerns several networks: The network Jim is connected to, the network Jane is connected to, and the set of networks in between them that are responsible for transmitting the message onwards. Let's say that the network characterization is a compound of them all. Jane's relevant context attributes are her location, activity, mood and social context.

The contextual information has impact on the trust Jane assigns to the message in the following way. First of all, Jim's being close to the site of the accident increases Jane's trust in the message. Jane can assume that he is an eyewitness of the accident. Secondly, his activity increases the trust as well. Jane knows that Jim works as a journalist for a newspaper and this gives Jane the impression that he is really trying to find out the true state of affairs in order to not report falsehoods. Third, Jane's being abroad increases her interest and trust in the message. If she was in her home town, she might well want to verify Jim's message from some news coverage. Finally, the long delay in receiving Jim's message— for example due to a technical network failure—decreases Jane's trust in the message. This is because the message content is in the present tense, but the fire might actually be already put out once Jane receives the message.

3 Context Information and Trustworthiness

3.1 Features of Context Information

As mentioned, our focus is on communication in context-sensitive environments. The smallest unit we consider to be enriched with trust values is one message sent between two communicating entities. Figure 1 depicts the concept of CONTEXT and how it is connected to MESSAGES. A message, concerning a TOPIC, is exchanged between SENDER and RECEIVER using a NETWORK. There are three kinds of contexts associated to a message, the context of the network used for transmitting the message (NETCTX), and

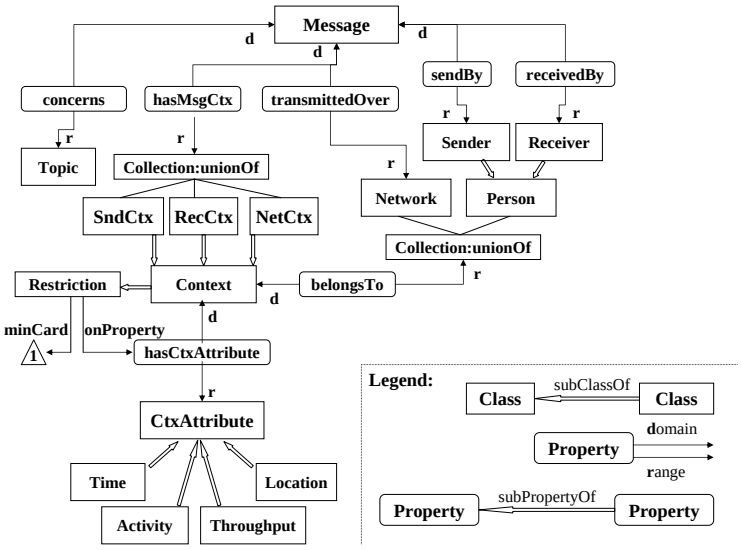

Fig. 1. OWL Ontology for attaching context information to messages

the contexts of the sender (SNDCTX) and receiver (RECCTX). In the following we refer to these contexts as ctx_n, ctx_s, and ctx_r, respectively. Context is described through CTXATTRIBUTES. Formally, ctx with *n* context attributes can be defined as follows:

$$ctx = \{ctxAttr(1), ctxAttr(2), \ldots, ctxAttr(n)\}$$

The specific attributes that constitute a user's context are addressed in several previous work and we do not go into details of that issue in this paper. Instead, we take a couple of context attributes that are frequently considered as relevant. Such are for example TIME, LOCATION and ACTIVITY. As for network, context can be expressed for example with network's quality of service (QoS) values such as THROUGHPUT [21]. Some of these attributes are included as examples in Figure 1. Restrictions on the kinds of context attributes for classes are needed, e.g., THROUGHPUT is a possible context attribute for ctx_n, but not for ctx_s or ctx_r. The BELONGSTO relation between the context and an entity (person or network), connects the message contexts with the associated sender, receiver and network. Appropriate ontological definitions (restrictions), assure that the sender of a message is the same entity as the entity to which the sender context of that message belongs. Moreover, appropriate restrictions on the range classes of the BELONGSTO property for three context subclasses assure that a sender context always belongs to a sender entity and so on. For reasons of space limitation, we do not show all restriction definitions of our ontologies. Instead, we refer the reader to http://www.csl.sri.com/users/denker/owl-sec/context/.

In Figure 2 we give an overview of the proposed trust ontology that extends topic-dependent trust and agent trust relations as proposed in [9], with the notion of context-sensitive trust in messages. The topic of a message can have impact on its trust level. For example, a geography student can have complete trust on his professor when it comes

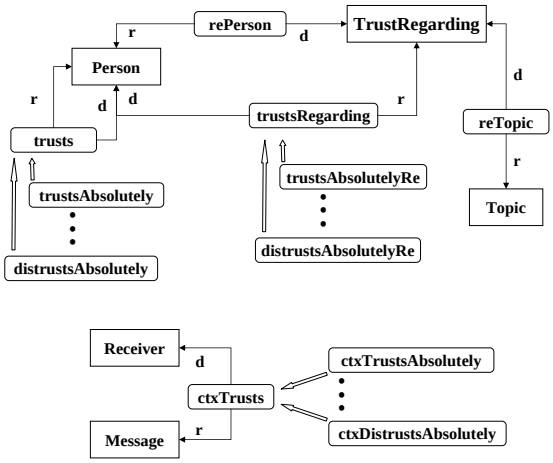

Fig. 2. Core concepts for context-sensitive trust

geographical phenomena, but much less trust in the area of cooking. This information is captured in the TRUSTSREGARDING property. Trust relationships between persons are captured by the TRUSTS relationship. Subproperties of these two relationships, depicted in Figure 2 with (TRUSTSABSOLUTELY,...,DISTRUSTSABSOLUTELY), and (TRUSTSABSOLUTELYRE,...,DISTRUSTSABSOLUTELYRE), conform with the nine levels of trust defined in [9, 8], namely TRUSTSABSOLUTELY, TRUSTSHIGHLY, TRUSTSMODERATELY, TRUSTSSLIGHTLY, TRUSTSNEUTRALLY, DISTRUSTSSLIGHTLY, DISTRUSTSMODERATELY, DISTRUSTSHIGHLY, and DISTRUSTSABSOLUTELY.

To capture the dependency of trust on context, we introduce the CTXTRUSTS relation between a message and its receiver. Since a message has a receiver, a sender and (possibly) a mediating network associated with it, which in turn have contexts attached to them, the value of the CTXTRUSTS relation will be computed using the TRUSTS and TRUSTSREGARDING relations, and context information. We envision the use of rules that describe how context-sensitive trust is computed on the basis of context-independent trust relations such as TRUSTS and TRUSTSREGARDING (See Section 4).

3.2 Accident Example Revisited

Let us now briefly return to the accident example presented above, but this time utilizing the concepts defined in the ontology. The contexts of our accident report example message 'The city hall is on fire!" are as follows. The message sender's, e.g. Jim's, context can be formalized as follows:

$$ctx_{Jim}^{fireMsg} = \{time^{fireMsg}, location^{fireMsg}, activity^{fireMsg}\}$$

with certain instances for the context attributes, such as $activity^{fi\ reMsg} = Journalist$. Network context, in turn, consists of the following:

$$ctx_{Net}^{fireMsg} = \{throughput^{fireMsg}\}$$

And finally, the receiver:

$$ctx_{Jane}^{fireMsg} = \{location^{fireMsg}, activity^{fireMsg}, mood^{fireMsg}, socCtx^{fireMsg}\}$$

The context values influence the overall trust value the receiver assigns to the message. This is defined using rules for reasoning about context-sensitive trust and functions for assigning context-sensitive trust values.

4 Rules and Functions for Context-sensitive Trust

On the basis of the context-sensitive message and the trust ontologies, we define rules that express how the value of the CTXTRUSTS relation is influenced by the values of context-sensitive relations. As a first simplified approach, rules are of the form:

```
[Rule label:]
IF expression
THEN ctxTrust_label(receiver,msg)=
     f(ctx-independent-trust(receiver,msg),factor)
```

A rule describes how certain context-sensitive information (captured in the expression) reduces or enhances the trust in a message, relative to its context-independent trust value. $Expression$ is a boolean combination of terms constructed of variables and predicates defined in our two ontologies. Though we will use an ad hoc notation in this paper, the example rules could easily be specified in a semantic rule notation such as SWRL [10] or Rei [13]. f is a function to compute context sensitive trust using trust factor $factor$. Before we go into details of the trust rules, we propose some possibilities for trust functions as well as functions to compute context-independent trust.

4.1 Trust Function for Context-sensitive Messages

In this section we consider context-independent trust that is computed using the topic-independent trust relations and the topic-dependent trust relations given in Figure 2 (cf. [9]). Topic-independent trust can be defined to extend the nine trust-levels as follows:

$$\textit{ctx-independent-trust}(r, msg) = \textit{trust}(r, s) + \textit{trustsRegarding}(r, tr)$$

where msg is a message with sender s, receiver r, concerning topic t and tr is a variable of type TrustRegarding with $rePerson(tr, r)$ and $reTopic(tr, t)$. Thus, the degree to which a receiver trusts a message independent of the message context is computed by adding the trust the receiver has in the sender of the message in general and more specifically concerning the topic of the message. This definition requires to define how trust relations are added. We propose the following simple definition, though other, more complex computations (e.g., using weights) could be defined if deemed necessary by the application. Moreover, if other context-insensitive trust relations are defined in a given application domain, then these could be included in a more complex definition of ctx-independent trust between a receiver and a message.

Under the assumption that all instances of the TRUSTS and TRUSTSREGARDING properties belong to one of the subclasses TRUSTSABSOLUTELY, ..., DISTRUTSABSOLUTELY, we define the following value assigning function[3]

$$\textit{value}\begin{pmatrix}\textit{distrustsAbsolutely}(r,s)\\ \textit{distrustsHighly}(r,s)\\ \dots\\ \textit{trustsNeutrally}(r,s)\\ \dots\\ \textit{trustsHighly}(r,s)\\ \textit{trustsAbsolutely}(r,s)\end{pmatrix} = \textit{value}\begin{pmatrix}\textit{distrustsAbsolutelyRe}(r,tr)\\ \textit{distrustsHighlyRe}(r,tr)\\ \dots\\ \textit{trustsNeutrallyRe}(r,tr)\\ \dots\\ \textit{trustsHighlyRe}(r,tr)\\ \textit{trustsAbsolutelyRe}(r,tr)\end{pmatrix} = \begin{pmatrix}-4\\ -3\\ \dots\\ 0\\ \dots\\ 3\\ 4\end{pmatrix}$$

Similarly, an inverse function value^{-1} is defined to assign context-independent trust predicates to integers, namely

$$value^{-1}\begin{pmatrix}-4\\ -3\\ \dots\\ 0\\ \dots\\ 3\\ 4\end{pmatrix} = \begin{pmatrix}\textit{cxt-independent-distrustsAbsolutely}(r,msg)\\ \textit{cxt-independent-distrustsHighly}(r,msg)\\ \dots\\ \textit{cxt-independent-trustsNeutrally}(r,msg)\\ \dots\\ \textit{cxt-independent-trustsHighly}(r,msg)\\ \textit{cxt-independent-trustsAbsolutely}(r,msg)\end{pmatrix}$$

With this, we can define

$$\begin{aligned}\textit{ctx-independent-trust}(r,msg) &= \textit{trust}(r,s) + \textit{trustsRegarding}(r,tr)\\ &= \textit{value}^{-1}(\lceil\frac{\textit{value}(\textit{trusts}(r,s)) + \textit{value}(\textit{trustsRegarding}(r,tr))}{2}\rceil)\end{aligned}$$

The context-insensitive trust is used to compute context-sensitive trust by applying a function that takes into account the trust factors determined by the rules. In the next section we illustrate these trust rules.

4.2 Rules for Computing Context-sensitive Trust

Next we show example rules for context-sensitive trust. The following rules formalize the implicit policies of the accident example presented in Section 2. The rules are defined for a variable `msg` of type message, and variables `s` of type sender, `r` of type receiver, `n` of type network, and `l` of type location. Moreover, we assume that additional predicates are defined for concepts in our ontology. For example, we will assume a boolean predicate CLOSE on locations, a CONTENT predicate for messages, and a LOCATION predicate for message content.

```
[Rule 1:]
IF sendBy(msg,s) AND  receivedBy(msg,r) AND
   hasMsgCtx(msg,ctx_s) AND belongsTo(ctx_s,s) AND
   hasCtxAtt(ctx_s,l) AND  close(l, location(content(msg)))
THEN ctxTrusts_1(r,msg)=f(ctx-independent-trust(r,msg), 2)
```

where, for example, f is defined as follows: For cit being one of the nine ctx-independent trust relations between receivers and messages (i.e., CXT-INDEPENDENT-ABSOLUTELYTRUSTS, ..., CXT-INDEPENDENT-ABSOLUTELYDISTRUSTS)

$$f(\textit{cit},\textit{factor}) = \textit{value}^{-1}(\lceil\frac{\textit{value}(cit) + \textit{factor}}{2}\rceil).$$

[3] The same function name (*value*) is used for both TRUSTS and TRUSTSREGARDING.

Thus, the rule says that context-sensitive trust is increased by two trust levels if the message is sent by someone close to the location about which the message reports. Similar rules could express that the receiver's trust is increased if the message talks about a location that is far from where the receiver resides when receiving the message.

```
[Rule 2:]
IF receivedBy(msg,s) AND  receivedBy(msg,r) AND
   hasMsgCtx(msg,ctx_r) AND belongsTo(ctx_r,r) AND
   hasCtxAtt(ctx_r,l) AND far(l, location(content(msg)))
THEN ctxTrusts_2(r,msg)=f(ctx-independent-trust(r,msg), 1)
```

A "0" trust factor in a rule does not influence the context-insensitive trust value, and negative trust factors decrease trust. As there are possibly more than one rule that apply to a (r, msg) pair, the overall trust value is given as

$$\mathit{ctxTrusts}(r, msg) = \frac{\sum_{\mathit{rules\ l}} \mathit{ctxTrusts\_l}}{\mathit{number\ of\ rules}} \tag{1}$$

As our example shows, predicates relating context information with message content are useful. We are currently developing our ontology to include predicates that we deem most common for the context attributes we consider. For example corresponding predicates for location and time on messages content will be useful to express rules. Other extensions of our ontology will be concerned with different kinds of trust. For example, a rule could express that a receiver of a message decreases the trust value of a message received, if the throughput of the network is very small and the message content large (e.g., including picture showing the town hall on fire). Though the receiver might generally trust the message content, she might doubt its temporal validity.

5 Related Work

Approach to reputation-based trust such as [9, 8] require explicit trust ratings. It depends on the definition of the trust rating to what extent contextual information is taken into account. REGRET [19, 20] is an approach for adding reputation in multi-agent systems. It can be said to come quite close to social contexts and their utilization in trust-assignment. However, it is limited to the social aspects of context-sensitivity. To our knowledge there is no general approach to include context information in trust ratings. Bizer et al [2] point out the importance to investigate context- and content-based trust policies on the Semantic Web. They very briefly introduce an architecture that has a query and trust evaluation layer to handle trust decisions, but give no further details. Trust, security, and privacy issues in context-aware environments can be approached in various ways. For example, people can be provided with control over who can access their personal information in various contexts as investigated in [7, 22].

Our approach defines a context and message ontology and extends well-known trust concepts into a context-sensitive trust ontology where trust is linked to message context. On the basis of this ontology, we can formalize rules that take context-sensitive information into account when computing trust values. However, our context-sensitive trust ontology could also be connected with other existing ontologies. For example,

SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) [3] could be used for expressing mobile contexts in a more detail, time ontology [18] for representing the temporal aspects, the network QoS ontology [21] for characterizing the network's properties, and FOAF (Friend of a Friend) [6] for expressing the contact details and other characteristics of people and their interrelationships.

6 Discussion

A relevant question to ask is who or what needs to provide trust functions and rules to enable context-sensitive trust calculation. If the burden is entirely put on the users themselves without any assistance from the information technology (IT) infrastructure, this approach might not be adopted. If, on the other hand, the trust is calculated either partially or entirely by the IT infrastructure, instantiating some of our trust relations, the adoption would be easier. Though one still needs to consider who in the value chain of operators in the Semantic Web would benefit from providing (semi)automatic trust calculations. A related question is whether the context information is automatically determined by the infrastructure or manually entered by the users, and does this have any significance to the trust people assign to messages.

Besides the "who", "how" is another question to be answered. How the context-sensitive trust factor is to be determined? Retrieving context information of users and other components of the system is difficult enough in the first place, but it should additionally be somehow transformed into meaningful factors of context-sensitive trust. This task encapsulates both a technological challenge of retrieving contextual information from text—such as Jim's being close to the city hall in the accident example above—and a broader challenge of determining the meaningful impact of the retrieved context information. In another kind of message, such as "I like the city hall's architecture", Jim's being in the immediate vicinity of the city hall is not so important. In this paper, we illustrated rules for transforming the context-information into a factor of trust, but did not address that the trust factor might be different for different aspects of trust, such as "trusting a message" vs. the more specific issue of "trusting the temporal validity of a message."

If the above "who" and the "how" questions get answered, one can still ask what purpose does the context-sensitive trust calculation serve. This is naturally related to the "who"-question, but is slightly different. Considering this "what purpose"-question entails reflecting the goal of the Semantic Web in general, i.e., the web of trust. The decentralized nature of the Semantic Web implies also decentralized trust, which brings about trust networks. In some cases the trust networks can vary based on the contextual information, as argued above, so the general purpose of the context-sensitive trust is to contribute to achieving the Semantic Web's general goal.

Additional complexity comes into play when reliability of context information becomes a parameter. In our current approach, we assume context information to be reliable. If we loosen this assumption, then context information needs to be evaluated for its trustworthiness. Finally, one needs to address the issue of disseminating the policies that constitute the relevant context information to trust determination for other agents in addition to message receivers.

7 Conclusions and Future Work

We presented ontologies to capture message context and context-dependent trust relations. We illustrated how rules can be used to define context-sensitive trust, and we proposed several trust evaluation functions.

Much work lies ahead and further investigations are necessary with respect to topics we did not address in this paper. For example, we intend to provide an implementation of context-sensitive trust rules and evaluation functions using SRI's rewriting framework Maude [16, 14, 4]. Rule declarations are at the heart of Maude and the Maude execution allows to define strategies for evaluating rule-based systems. This way we can automate the process of trust evaluation, as well as experiment with different kinds of trust functions, as new functions can be added easily. Maude is a very efficient, fast rewrite engine with thousands of rewrites per second. Thus, experimentation with a large number of rewrite rules and facts to evaluate the effects of various trust functions can be handled in this framework.

Future work will also address experiments evaluating the adequacy of our trust ontology by illustrating its use in various application domains. On one hand, it will be important to exploit existing ontologies, such as those developed in the DAML project [5] for time and space, and other application specific ontologies. On the other hand, experiments have to be performed the compare results with and without the use of context information for trust determination.

Assigning context-dependent trust on messages presupposes extracting relevant information from the message contents (such as the location of the city hall) and comparing it with the contexts of the sender, network, and the receiver. Information extraction from free-form messages like "The city hall is on fire!" is by no means an easy task. Instead, if the message contents were structured and conformed to some machine-understandable semantics, the information extraction and combination with context attributes would become much easier. In our future work we are taking steps towards this direction by analyzing trust people assign in structured service descriptions, such as pizzeria recommendations and bus timetables. These service descriptions would be passed to the users both directly from the service providers and also—and more importantly, as far as our research is concerned—via other users such as their friends or colleagues. The information extraction would likely become easier, since the pizzeria recommendation, for example, would contain the location of the pizzeria and its opening hours, which could be mapped with the contexts of the message sender and receiver.

Acknowledgements. We would like to thank the anonymous referees for their helpful comments and for suggesting interesting questions, e.g., reliable context information and dissemination of context policies, for future research.

References

1. B. Acrement. Elements for building trust. Available at: http://www.imakenews.com/smei/e_article000051474.cfm.
2. C. Bizer. Using context- and content-based trust policies on the Semantic Web. In *Poster in Proceedings of WWW 2004, May 17-22, 2004, New York*, pages 228–229. ACM, 2004.

3. H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, August 2004.
4. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. *Maude: Specification and Programming in Rewriting Logic*. SRI International, Computer Science Laboratory, Menlo Park, CA, January 1999. Available at: http://maude.csl.sri.com/manual.
5. Darpa agent markup language (daml) project web site. Available at: http://www.daml.org.
6. Friend of a friend (foaf) project web site. Available at: http://www.foaf-project.org.
7. F. Gandon and N. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):241–260, 2004.
8. J. Golbeck and J. Hendler. Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW) 2004*, Northamptonshire, UK, October 2004. Springer.
9. J. Golbeck, B. Parsia, and J. Hendler. Trust networks on the semantic web. In M. Klusch, S. Ossowski, A. Omicini, and H. Laamanen, editors, *Proceedings of Cooperative Intelligent Agents (CIA) 2003*, pages 238–249, Helsinki, Finland, August 2003. Springer.
10. I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. Swrl: A semantic web rule language combining owl and ruleml. Technical Report Version 0.5, November 2003. Available at: http://www.daml.org/2003/11/swrl.
11. J. Hradesky. *Total Quality Management Handbook*. McGraw Hill Text, December 1994.
12. iTrust - an information society technology (ist) working group. Available at: http://www.itrust.uoc.gr.
13. L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2002.
14. Maude Web site. Available at: http://maude.csl.sri.com/, 2000.
15. D. McGuinness and F. van Harmelen. Owl web ontology language overview. The World Wide Web Consortium (W3C), February 2004. Available at: http://www.w3.org/TR/owl-features.
16. J. Meseguer. A logical theory of concurrent objects and its realization in the Maude language. In G. Agha, P. Wegner, and A. Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming*, pages 314–390. The MIT Press, 1993.
17. T. Moran and P. Dourish. Special issue on context-aware computing. *Human Computer Interaction*, 2001.
18. F. Pan and J. Hobbs. Time in owl-s. In *Proceedings of the 2004 AAAI Spring Symposium on Semantic Web Services*, Stanford, CA, March 2004.
19. J. Sabater and C. Sierra. Regret: reputation in gregarious societies. In *Proceedings of the fifth international conference on Autonomous agents*, pages 194–195. ACM Press, 2001.
20. J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 475–482. ACM Press, 2002.
21. S. Toivonen, H. Helin, M. Laukkanen, and T. Pitkäranta. Context-sensitive conversation patterns for agents in wireless environments. In S.K. Mostefaoui, Z. Maamar, and O. Rana, editors, *Proceedings of the The First International Workshop on Ubiquitous Computing (IWUC'04) held in conjunction with ICEIS 2004*, pages 11–17, Porto, Portugal, April 2004. INSTICC Press.
22. S. Toivonen, J. Kolari, and T. Laakko. Facilitating mobile users with contextualized content. In *Proceedings of the Artificial Intelligence in Mobile Systems (AIMS) 2003 Workshop held in conjunction with Ubicomp 2003*, Seattle, WA, October 2003.

Personalized Reputation Management in P2P Networks

Paul - Alexandru Chirita[1], Wolfgang Nejdl[1], Mario Schlosser[2], and Oana Scurtu[1]

[1] L3S Research Center / University of Hannover
Deutscher Pavillon Expo Plaza 1
30539 Hannover, Germany
{chirita,nejdl,scurtu}@l3s.de
[2] McKinsey & Company Inc. / Stanford University
schloss@db.stanford.edu

Abstract. P2P networks have become increasingly popular in the recent years. However, their open, distributed and anonymous nature makes them very vulnerable against malicious users who provide bad responses to requests from other peers. Motivated by this observation, various solutions for distributed reputation systems have been presented recently. In this paper, we describe the first reputation system which incorporates both user-individual personalization and global experiences of peers in the network, for the distributed computation of reputation values. We also present a secure method to compute global trust values, thus assuring identification and isolation of malicious peers. Finally, our simulations show that our system is robust even against attacks from groups of malicious peers deliberately cooperating to subvert it.

1 Introduction

P2P networks are powerful distributed infrastructures allowing any peer to search for and offer content and services. They are capable of handling enormous amounts of resources while maintaining an organized and balanced topology. However, because of their open nature, they also require more complex reputation schemes, as malicious users can introduce corrupt data and/or harmful services much easier than in centralized systems. Such reputation algorithms are usually based on aggregating the trustworthiness information collected by each peer with that of several or all other peers, thus generating a micro or macro *web of trust*.

As the size of current P2P networks is continuously increasing, recent research has concentrated on designing more personalized trust management algorithms, while also addressing their robustness against attacks of malicious peers. Some of these techniques employ only local computations on sub-graphs of the entire P2P network [15, 9]. Their output is personalized at the cost of not considering global experiences of peers. [11] tackles this, but only for statements from the Semantic Web. Finally, techniques which include experiences of all peers [8] do not address the issue of personalization.

In this paper we introduce a novel algorithm for computing personalized reputation values in P2P networks and test its robustness against several possible attacks in a simulated P2P environment. We use a real-world experimental setup simulating power-law distributions on peer links as well as content distribution, as can be observed in many current P2P networks.

We start with a short introduction of algorithms related to our research in Section 2. The distributed computation of personalized reputation values is introduced in Section 3, and then extended into a secure version in Section 4. Section 5 presents our experimental results, section 6 contains our conclusions.

2 Background and Previous Work

2.1 Trust and Reputation

Although the area of reputation algorithms has received increased attention recently, some issues are still left open. [15] gives a very good overview over existing approaches. The paper contains the first categorization of trust metrics and a definition of trust elements (model, metrics, etc.) in the Semantic Web. Its main contribution is *Appleseed*, a fixed-point personalized trust algorithm inspired by spreading activation models. Similarly, an important aspect for us is the introduction of "backward trust propagation" (i.e. virtual edges from every visited node x to the computation seed node s), which solves several rank normalization problems (e.g. distinguish between a peer with whom some peer i did not interact and a peer with whom i had bad experiences).

[11] builds a Web of trust, with each user having to maintain trust values on a small number of other users. The algorithm presented is designed for an application within the context of the Semantic Web, composed of logical assertions. This helps introducing personalization, as each user would have a level of local belief in statements and a level of trust in other users, all of which could then be merged to reflect a global meaning. [9] presents an interesting method based on a quorum of other peers, who are asked about their opinion on some peer p, instead of relying on a fixed-point algorithm. This results in reduced network traffic, but comes at the cost of not achieving a global perspective on the trustworthiness of peer p. A related approach is presented in [5], where the FOAF [3] schema is extended to contain trust assertions between peers. The inferred rating from a source peer to a sink one is (recursively) computed using the weighted average of the neighbors' reputation ratings of the sink. [8] is a fixed-point PageRank-like distributed computation of reputation values in a P2P network. We used its model in designing our algorithm, as well as the investigations about possible attacks from malicious peers. Just as [7] improves [10], our algorithm extends the capabilities of [8] by introducing personalization into the computation.

For global ranking algorithms, we distinguish three targets based on fixed-point iterations: (1) Web pages [10, 7, 6], (2) linked documents in P2P networks [12, 14, 1], and (3) peers in P2P networks [8]. In all cases, input data can be represented as a directed graph with the targets of the algorithm as nodes. For ranking Web pages or documents, graph edges are the hyperlinks, whereas for building reputation values they resemble peers' experiences with other peers. These approaches are summarized in table 1.

2.2 Personalized PageRank

Description. [7] is the most recent investigation towards personalized page ranks. We give a more detailed description of this algorithm in the following paragraph, as we will extend this algorithm to compute personalized trust values in a distributed way.

Outcome	Node in graph	Edge in graph
Web page ranks [10]	Web page	Hyperlink
Personalized Web page ranks [7, 6]	Web page	Hyperlink
Document ranks (distributed) [12, 14]	Document on a peer	Hyperlink between two documents
Personalized document ranks (distributed) [1]	Document on a peer	Hyperlink between two documents
Reputation values (distributed) [8]	Peer	Download experience of peers
Personalized reputation values (distributed) - this paper	Peer	Download experience of peers

Table 1. Hyperlink structures used by different ranking algorithms

[7] introduces personalized PageRank Vectors (PPV) computed for each user. The personalization aspect of this algorithm stems from a *set of hubs* (H), and each user has to select her *preferred pages* from this set. PPVs can be expressed as a linear combination of basis vectors (PPVs for preference vectors with a single non-zero entry corresponding to each of the pages from P, the preference set), which could be selected from the precomputed basis hub vectors, one for each page from H. To avoid the massive storage resources basis hub vectors would use, they are decomposed into partial vectors (which encode the part unique to each page, computed at run-time) and the hub skeleton (which captures the interrelationships among hub vectors, stored off-line).

Algorithm. In the first part of the paper, the authors present three different algorithms for computing basis vectors: "Basic Dynamic Programming", "Selective Expansion" and "Repeated Squaring". In the second part, specializations of these algorithms are combined into a general algorithm for computing PPVs, as depicted below.

Algorithm 1. Personalized PageRank in a centralized fashion.

Let $D[p]$ be the approximation of p's basis vector, and $E[p]$ the error of its computation.
1.(Selective Expansion) Compute the partial vectors using
$Q_0(p) = V$ and $Q_k(p) = V \setminus H$, for $k > 0$, in the formulas below:

$\mathbf{D_{k+1}[p]} = \mathbf{D_k[p]} + \sum_{q \in Q_k(p)} c \cdot E_k[p](q)\mathbf{x_q}$

$\mathbf{E_{k+1}[p]} = \mathbf{E_k[p]} - \sum_{q \in Q_k(p)} E_k[p](q)\mathbf{x_q} + \sum_{q \in Q_k(p)} \frac{1-c}{|O(q)|} \sum_{i=1}^{|O(q)|} E_k[p](q)\mathbf{x}_{O_i(q)}$

Under this choice, $D_k[p] + c * E_k[p]$ will converge to $\mathbf{r_p} - \mathbf{r_p^H}$,
the partial vector corresponding to page p.

2.(Repeated squaring) Having the results from the first step as input, one can now compute the hubs skeleton ($r_p(H)$). This is represented by the final $D[p]$ vectors calculated using $Q_k(p) = H$ into:

$\mathbf{D_{2k}[p]} = \mathbf{D_k[p]} + \sum_{q \in Q_k(p)} E_k[p](q) * D_k[q]$

$\mathbf{E_{2k}[p]} = \mathbf{E_k[p]} - \sum_{q \in Q_k(p)} E_k[p](q)\mathbf{x_q} + \sum_{q \in Q_k(p)} E_k[p](q)E_k[q]$

---

3. Let $u = \alpha_1 p_1 + \cdots + \alpha_z p_z$, $p_i \in H$, $i = 1, 2, .., z$, be a preference vector, and let:
$r_u(h) = \sum_{i=1}^{z} \alpha_i (r_{p_i}(h) - c * x_{p_i}(h)),\ h \in H$, computable from the hubs skeleton.
The PPV v for u can then be constructed as:
$v = \sum_{i=1}^{z} \alpha_i (r_{p_i} - r_{p_i}^H) + \frac{1}{c} \sum_{h \in H\ r_u(h) > 0} r_u(h) * \left[(\mathbf{r_h} - \mathbf{r_h^H}) - c * x_h\right]$

---

3 Personalized Reputation Values in P2P Networks

3.1 Introduction

In our distributed algorithm for computing personalized reputation values we start from a set H of pre-trusted peers (called *hub peers* hereafter). Each peer will have its own preference set $P \subset H$. Even though all hub peers are highly trusted, each peer trusts some of them (those from P) more than it trusts the others. Generally, this happens because they provide better quality of service or faster and more downloads, in a specific area of interest. In a services network, for example, a peer might prefer the abilities of one or several hub peers, because they supply very good results in the specific service domain it is interested in.

What is the intuition behind personalized trust values in a P2P network, and, even more importantly, in which way does personalization of trust values improve on global trust rating systems, where only one trust value is established per peer, encompassing all other peers' interactions with this peer? The notion of personally selected pre-trusted peers gives an elegant answer: When a peer in the network selects a subset of pre-trusted peers which it trusts most, it does not necessarily consider these peers as more trustworthy than other pre-trusted peers - in fact, all pre-trusted peers should be entirely trustworthy, i.e., always strive to provide authentic file uploads or non- malicious services. Rather, a peer selects those pre-trusted peers whose trust ratings and experiences with peers in the network are most *relevant* to this peer's operations within the P2P network. The peer may operate in a content domain in which certain peers have provided seamless and perfectly trustworthy service - even though they would have a low global trust rating due to other peers being much more active in responding to popular queries. Hence, personalization of trust values does not only establish a web of trust, it creates personal webs of trust in which peers with similar interests cooperate to choose the most trustworthy peers in their peer group.

But what if a peer does not prefer *any* hub peer, but some other peer h'? There are several solutions which cope with this (see [2]), the simplest one being a breadth-first search around h' with the minimum radius such that at least a hub peer is discovered, thus selecting the hub peer(s) closest to h'.

We divide the algorithm in three parts, as presented in section 2.2: One part focuses mostly on peers outside the set of pre-trusted peers, $V \setminus H$ (Algorithms 3.1.1 and 3.1.2), one on peers within the set of pre-trusted peers H (Algorithm 3.2), and the final algorithmic step ties everything together (Algorithm 3.3).

3.2 Partial Vectors

This part of the algorithm consists of one special initialization step and several succeeding steps. Even though its focus is on peers from $V \setminus H$, peers $p \in H$ also gather their components of $\mathbf{D}$ and $\mathbf{E}$. All peers normalize their trust values as $\gamma_q(i) = \gamma_q(i) / \sum_i \gamma_q(i)$. In the first step, each peer $q \in V$ computes $E[p](q)$. As peers $p \in H$ are known, each peer $q \in V$ can set its initial values $D_0[p](q)$ and $E_0[p](q)$ by itself to:

$$D_0[p](q) = \begin{cases} c \ , q \in H \\ 0 \ , otherwise \end{cases} ; E_0[p](q) = T_0[p](q) = \begin{cases} 1 \ , q \in H \\ 0 \ , otherwise \end{cases} \quad (1)$$

Non-hub peers. After this initialization, the following operations will be executed in parallel by each peer $q \in V \setminus H$ for each $p \in H$:

Algorithm 3.1.1. Distributed computation of partial vectors by peers in $V \setminus H$.

1: Send $\frac{1-c}{|\mathbf{O(q)}|} \cdot T_k[p](q) \cdot \gamma_q(i)$ to all peers i from which q has downloaded files, including those from H.
2: Wait from all peers which downloaded files from p their $T_k[p](*)$ (at this step k)
3: After all $T_k[p](*)$ values have been received, compute $T_{k+1}[p](q)$ as: $T_{k+1}[p](q) = \sum_{v \in \mathbf{I(q)}} T_k[p](v)$
4: Compute:
$D_{k+1}[p](q) = D_k[p](q) + c \cdot T_k[p](q)$
N being the number of steps we apply the Selective Expansion algorithm.
5: If there are more iterations left, go to 1
6: Each peer $q \in V \setminus H$ computes $(r_p - r_p^H)(q) = D_N[p](q) + c \cdot T_N[p](q)$, its component of the partial vector corresponding to p.

Theorem 1. *In the distributed version of the Selective Expansion algorithm, for $p \in H$ each peer $q \in V \setminus H$ has to compute:*

$$T_{k+1}[p](q) = \sum_{v \in \mathbf{I(q)}} \frac{1-c}{|\mathbf{O(v)}|} \cdot E_k[p](v) \cdot \gamma_v(q) \tag{2}$$

Proof. Due to space limitations, we refer the reader to [1] for the proof of this theorem.

Hub peers. In the special first step, a peer $p \in H$ will send $\frac{1-c}{|\mathbf{O(q)}|} \cdot T_k[p](p) \cdot \gamma_p(i)$ to all peers i from which it has downloaded files, including those from H. After that, it will execute the following operations:

Algorithm 3.1.2. Distributed computation of partial vectors by peers in H.

1: Wait from all peers which downloaded files from p their $T_k[p](*)$ (at this step k)
2: After all $T_k[p](*)$ values have been received, do $T_{k+1}[p](p) = \sum_{v \in \mathbf{I(q)}} T_k[p](v)$
3: If there are more iterations left, go to 1
4: Compute: $E[p](p) = \sum_{k=1}^{N} T_k[p](p)$
5: Set $D_N[p](p)$ to c
6: Each peer $p \in H$ computes $(r_p - r_p^H)(p)$, its component of its partial vector.

Algorithms 3.1.1 and 3.1.2 perform a power-iteration, and they would therefore converge also in the presence of loops in G (see [7, 10] for details), whereas the high level of dynamics of a P2P network (peers skipping participation in one or more iterations because entering or leaving the network, etc.) would only result in some additional iterations until convergence.

3.3 Hub Skeleton

In the second phase of the algorithm, each hub peer ($p \in H$) has to calculate its hub skeleton ($\mathbf{r_p}(\mathbf{H})$) using as input the results from the previous stage. The output is stored in the values $\mathbf{D_{2k}}[\mathbf{p}]$ obtained after the last iteration of the following operations:

Algorithm 3.2. Distributed computation of hub skeleton (only in H).

1: Calculate $\mathbf{D_{2k}}[\mathbf{p}]$ and $\mathbf{E_{2k}}[\mathbf{p}]$, using the centralized version formulas
2: Multicast the results to all other peers $q \in H$, possibly using a minimum spanning tree for that.
3: If there are more iterations left, go to 1.
4: Every hub peer broadcasts its $\mathbf{D_{2N}}[\mathbf{p}]$ sub-vector (only the components regarding pages from H).

As this step refers to hub-peers only, the computation of $\mathbf{D_{2k}}[\mathbf{p}]$ and $\mathbf{E_{2k}}[\mathbf{p}]$ can consider *only* the components regarding pages from H.

3.4 PPV

As the $\mathbf{D_{2N}}[\mathbf{p}]$ sub-vectors have been broadcast, *any* peer $v \in V$ can now determine its $\mathbf{r_v}(\mathbf{P})$ locally, using the original formula (see sec. 2.2). It can also calculate its partial PPV containing its reputation and the reputation of any other peers from its own point of view. If *NB* is the set of peers whose rank v wants to compute, it must do the following:

Algorithm 3.3. Computation of the Personalized Reputation Vector.

1: Request the components of $\mathbf{r_p} - \mathbf{r_p^H}$ for all $p \in H$ from all peers from NB.
2: Compute the components of the PPV using the original formula (see section 2.2).

Of course, if later v wants to compute the reputation value of another peer, it would only need to ask the new peer about its components of $\mathbf{r_p} - \mathbf{r_p^H}$.

4 Secure Computation of Personalized Reputation Values

From a security point of view, the algorithm as described above contains two critical components. First, pre-trusted peers play an important role in computing trust values. Hence an implementation of the algorithm has to make sure that they behave correctly. Fortunately, only very few pre-trusted peers are required in a network (see also Section 5) such that the administrative task of ensuring this behavior will present little overhead. Second, non-pre-trusted peers calculate trust values for themselves, or at least contribute to their calculation. This gives each peer major power over his own trust rating. A secure

version of the algorithm thus has to ensure that malicious peers in the network have limited or no impact on the computation of their own trust ratings.

We achieve this in two steps: First, by having another, deterministically chosen peer in the network take over a peer's calculation job on his own trust value, becoming this peer's proxy calculation peer. Second, by adding redundancy to the calculations through having each calculation being performed by several peers in parallel. Each peer can be queried for the results of his calculations, and the final result of a calculation is determined through a majority vote among the collaborating peers.

We organize the network into a distributed hash table, using a scheme such as Chord [13]. Each peer's IP address is mapped into a logical hash space. Through random assignment upon joining the network, each peer covers a part of the hash space, becoming the proxy calculation peer for all peers whose network address has been mapped into its domain in the hash space. Several proxy peers can be associated with a peer by applying several different hash functions. Each hash function is deterministic, i.e., each peer in the network can infer the location of a proxy calculation peer on the DHT grid and route requests for calculation results appropriately.

5 Experimental Results

We tested our algorithms on simulated P2P networks which exhibit a power-law connectivity distribution, because popular real-world networks (e.g. Gnutella [4]) are structured as such. Peers were selected as download source with a probability proportional to their rank: Due to the power-law distribution, highly ranked peers were selected much more often than other peers, which reflects real-world P2P traffic distributions. Unless stated differently, the set of pre-trusted peers contained 5 (hub) peers, and the preference set of each ordinary peer contained 2-3 hub peers randomly selected. We always ran algorithms 3.1.* for 6 iterations, and algorithm 3.2 for 10 iterations, as in [7].

Resources used. We first generated a graph with 10,000 nodes in order to assess the amount of network traffic required by the algorithm. The set of pre-trusted peers contained 150 (hub) peers, while the preference set of each ordinary peer contained 30 hub peers randomly selected. Results are summarized in tables 2 and 3.

In our statistics, we distinguish between ordinary peers and hub peers. Hub peers generate significantly more traffic because of the second phase of the computation (algorithm 3.2), in which they need to multicast their intermediate results to all other hub peers. However, there is usually a small number of such peers compared to the size of the network, and they also control more resources (e.g. bandwidth or processing power).

	Max. Size	**Avg. Size**
All Data	1,989,072	1,979,695
$\neq 0$ **Data**	1,944,764	1,938,796

Table 2. Data transferred by hub peers (nb. of real values sent)

	Max. Size	**Avg. Size**
All Data	101,232	4,231.3
$\neq 0$ **Data**	5,360	84.69
All Data, 1 iteration	665	27.91
$\neq 0$ **Data, 1 iteration**	171	2.043

Table 3. Data transfered by ordinary peers (nb. of real values sent)

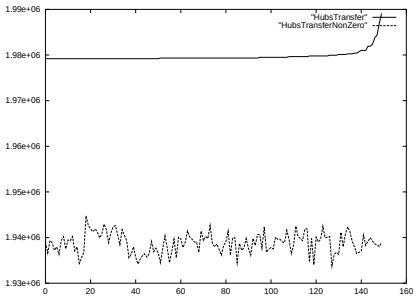

Fig. 1. Data transfered by hub peers (nb. of real values sent)

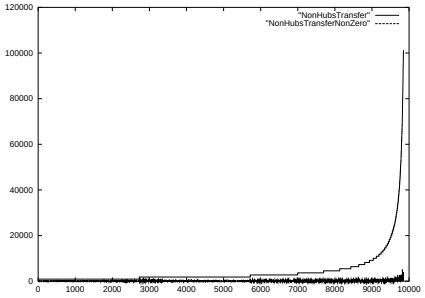

Fig. 2. Data transfered by ordinary peers (nb. of real values sent)

We observed only reduced communication for ordinary peers, which means that for the majority of the network, the computation needs only very small bandwidth. We also noticed that many peers often send the value ”0” through the network (as a correct intermediate value), and therefore a further modification of the algorithm which avoids sending it would reduce the traffic even more. Figures 1 and 2 present the traffic generated by each peer in the network. As we are dealing with a power-law network, we can observe the power-law distribution of data size among the ordinary peers. This is because the more neighbors a peer has, the more data it needs to exchange.

Robustness. An important aspect of a reputation algorithm is to be robust against attacks of malicious peers, which try to subvert the network by uploading inauthentic files or providing faulty services. For these tests, we simulated possible attacks as described in [8]. As our algorithm is also based on a power-iteration, the percentages of inauthentic files malicious peers are able to upload in the presence of the reputation system should be similar to those from [8]. In all experiments, we assumed the network to have an initial structure, i.e. peers have some initial trusts in other peers, generated randomly following a power-law distribution. We ran 30 query cycles, each of them consisting of sending 50 queries through the network. After each query cycle one more iteration of the selective expansion took place, as well as an update of the reputation vectors at each peer. The results of repeated squaring were updated only once in 5 cycles, as they need more computational resources.

Threat Model A. Malicious peers always provide an inauthentic file when selected as download source. They set their local trust values to 1 - s_{ij}, i.e. the opposite of their real trust value. The network consists of 63 good peers and 0, 7, 14, 25, 37, and 60 malicious peers respectively. Each good peer answers a corrupt file with 5% probability.

Threat Model B. Malicious peers of type A form a malicious collective. They set their local trust values to 1 - sij for good peers, and to 1 for any other malicious peers (i.e. complete trust in the other malicious peers). The peer distribution has the same characteristics as in threat model A.

Discussion. In both cases, the average fraction of inauthentic downloads is about 11% and does not exceed 13.5%, which represents a significant decrease against a net-

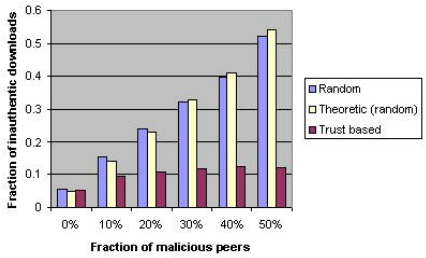

Fig. 3. Threat A: Mal. peers provide corrupt files and set their trust opposed to the real values.

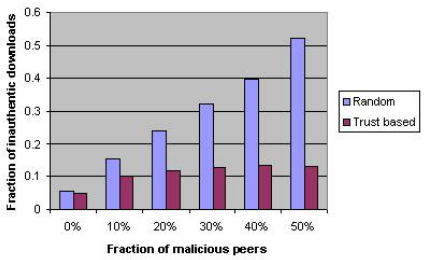

Fig. 4. Threat B: Malicious peers of type A also form a collective.

work without a reputation system in place. Moreover, malicious collectives are broken by our algorithm, as they only add 1-2% extra inauthentic downloads.

Threat Model C. Malicious peers of type B provide an inauthentic file in $f\%$ of all cases when selected as download source, with an f varying from 0 to 100 in steps of 10. The network consists of 53 good peers and 20 malicious ones.

Threat Model D. A first group of malicious peers of type D provide authentic files with 95% probability (just as the good peers), but additionally assign a trust of 1 to all malicious peers of the second type, B. There are 63 good peers and 40 malicious ones, the latter ones having different roles in different experiments, as depicted in figure 6.

Discussion. Here, malicious peers try to increase their rating by also providing good files. In model C, the maximum amount of inauthentic files they insert in the network is 17.6%, achieved when providing 50% good answers. For $f = 70\%$ (e.g. 365 good answers and 1215 bad ones), there were only 7.3% corrupt downloads in the entire network. Finally, the increase of bad downloads is also small when some peers acting correctly are trying to boost the reputation of the malicious ones. Although results in the right side of figure 6 are comparable to the random case, they require too much effort from malicious peers. For example, with 15 B peers and 25 D peers, they need to upload 1420 good files altogether in order to distribute 1197 inauthentic ones.

Generally, we can conclude that the robustness of our algorithm against malicious peers is very similar to that of EigenTrust [8] and thus the addition of personalization into a fixed-point reputation algorithm does not make it more vulnerable.

6 Conclusions

We have presented an algorithm which computes personalized global reputation values in P2P networks based on a fixed-point iteration and having peers' previous experiences as input. We also described a secure method to compute global trust values, in order to assure identification and isolation of malicious peers. We showed how to implement the reputation system in a scalable and distributed manner and simulated several possible subversion attacks. Our algorithm proved to be as robust against them as [8], while adding personalization to the global ranks computed by each peer.

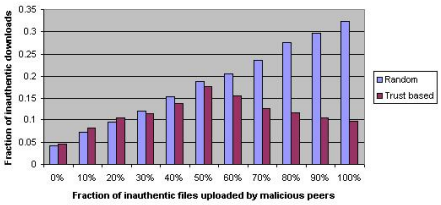

Fig. 5. Threat C: Malicious collective providing f% correct answers.

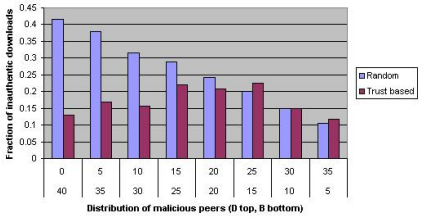

Fig. 6. Threat D: Malicious group boosting the reputation of a malicious collective of type B.

References

1. Paul-Alexandru Chirita, Wolfgang Nejdl, and Oana Scurtu. Knowing where to search: Personalized search strategies for peers in p2p networks. In *Proceedings of the P2P Information Retrieval Workshop held at the 27th International ACM SIGIR Conference*, 2004.
2. Paul-Alexandru Chirita, Daniel Olmedilla, and Wolfgang Nejdl. Pros: A personalized ranking platform for web search. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Aug 2004.
3. E. Dumbill. Finding friends with xml and rdf.
4. Gnutella web page: http://www.gnutella.com/.
5. J. Golbeck, B. Parsia, and J. Hendler. Trust networks on the semantic web. In *Proceedings of Cooperative Intelligent Agents*, 2003.
6. T. Haveliwala. Topic-sensitive pagerank. In *In Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii*, May 2002.
7. G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International World Wide Web Conference*, 2003.
8. S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th Intl. WWW Conference*, 2003.
9. S. Marti and H. Garcia-Molina. Limited reputation sharing in p2p systems. In *Proceedings of ACM Conference on Electronic Commerce (EC04)*, 2004.
10. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.
11. M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of the 2nd International Semantic Web Conference*, 2003.
12. K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed pagerank for p2p systems. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, 2003.
13. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup lervice for internet applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, USA, August 2001.
14. A. Yamamoto, D. Asahara, T. Itao, S. Tanaka, and T. Suda. Distributed pagerank: A distributed reputation model for open peer-to-peer networks. In *Proceedings of the 2004 Symposium on Applications and the Internet-Workshops*, 2004.
15. C. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service*, 2004.

Trust Strategies for the Semantic Web

Kieron O'Hara, Harith Alani, Yannis Kalfoglou, and Nigel Shadbolt

Intelligence, Agents, Multimedia Group, School of Electronics and Computer Science, University of Southampton, UK
{kmo, ha, y.kalfoglou, nrs}@ecs.soton.ac.uk

Abstract. Enabling trust on the Semantic Web to ensure more efficient agent interaction is an important research topic. Current research on trust seems to focus on developing computational models, semantic representations, inference techniques, etc. However, little attention has been given to the plausible trust strategies or tactics that an agent can follow when interacting with other agents on the Semantic Web. In this paper we identify five most common strategies of trust and discuss their envisaged costs and benefits. The aim is to provide some guidelines to help system developers appreciate the risks and gains involved with each trust strategy.

1 Introduction

Trust is at the heart of the Semantic Web (SW) vision. Trust is a method of dealing with uncertainty; when dealing with independent agents, institutions or providers of resources (including knowledge), one trusts them if one accepts their characterisation of what they will do. Trust can be a moral notion (X trusts Y to act in X's interests), or not (X trusts Y to perform some task T). Adopting the attitude of trust towards others means that one can plan and cooperate more efficiently, at the cost of greater risk of wasting resources when trust is misplaced.

The SW, conceived as a *collection* of agents, will therefore function more effectively when trust is licensed. As pointed out in [2], trust is essential for agents collaboration; each agent will have to make subjective trust judgements about other agents with respect to the services they claim to be able to supply.

There has been a good deal of research in this field [19]. But from the SW point of view, there are some interesting higher-level problems still to be addressed [26]. In the first place, many of the approaches are firmly in the field of multi-agent systems, where trust is also clearly a deep issue. However this has meant that some of the important problems specific to the SW are inevitably glossed over. Furthermore, many approaches have been technical 'fixes' which make a good deal of sense in circumscribed, specific contexts, such as commercial negotiations exploiting some relatively rigidly defined notion of 'agent'.

Secondly, because of the technical nature of many solutions, there are few signs of consensus emerging in the field. In itself, this is not a serious problem - the more solutions, even partial ones, the better, for heterogeneous and distributed systems. But we argue in this paper that some higher-level patterns are emerging, and outlines of general approaches becoming visible, which may be of help in understanding and comparing current research, and planning future developments. These higher-level patterns can be understood as *strategies* for placing trust. Or, in other words, general attitudes towards other agents in conditions of uncertainty. The best strategy will vary based on many variables, such as context, risk, cost, task undertaken, etc.

2 Situating Trust for the Semantic Web

There are a number of challenges to set an infrastructure that could promote trust, not least the lack of consensus in the field as to the best way of doing it [26]. How should trust be modelled? What information *is* relevant? What features should online trust have? Is it analogous to offline trust? How do we preserve the link between trust of other agents, and the incentives for those agents to be trustworthy?

Let us briefly set out some of the more pertinent challenges. To begin with, the SW, as a collection of agents, will be highly distributed. In such a structure, the idea of a centralised authority is hard to sustain, though such authorities are useful for fostering trust. More plausible is the idea of various competing authorities in restricted domains. Against that is the idea of an agent being responsible for gathering enough information for its own trust judgements; this would be ideal in a large distributed system, but may be prone to error, depending on how many information sources there were and how reliable they were [6][4].

Second, assuming that trust on the SW is not totally centralised, an agent will have to be able to discover relevant information from a variety of heterogeneous sources. [11] suggests that an agent must combine information from various sources effectively (eg from its own experience and that of other agents, agents certificates, and agents's roles), estimate the trustworthiness of those sources, and manage to cope with strategies of lying agents.

Third, agents must be able to exchange information effectively [14]. In particular, it must be possible to bootstrap trust in a context before there have been enough transactions between the protagonists to make firm judgements.

Fourth, how can we model trust without undermining incentives for trustworthiness (if an agent knows what signals I look for when judging trust, that agent now only has an incentive to give out those signals, not to behave in a trustworthy manner)? What knowledge should agents try to gather? And what properties of trust need be taken into account?

Fifth, as we have conceived the SW, trust is subjective and dependent on context [4]; an agent may be trusted to do one task with one set of resources, yet not trusted to do a different task with different resources. Will it be possible to produce models of trust that respect this context-sensitivity without increasing computational overheads too greatly?

In the next two sections, we will look at the costs and benefits of certain strategies of placing trust, and consider which are well-placed to address the challenges we have outlined in this section.

3 Trust strategies for the Semantic Web

A strategy for trust is an attitude towards the relative costs and benefits of secured interaction, unsecured interaction or no interaction at all. For instance, a safety critical system may regard avoiding catastrophic failure as an important part of its function, even if the risk is very low. Hence it will be prepared to accept high costs for refusing cooperation with other agents where failure could lead to a catastrophe. On the other hand, an agent might accept a broker's suggestion of an unreliable SW service if the cost of failure was low.

There are many strategies possible for dealing with trust on the SW. Examination of the SW and related fields leads us to sketch five potential strategies, based on a rough division of approaches. We identify five basic strategies (fig. 1).

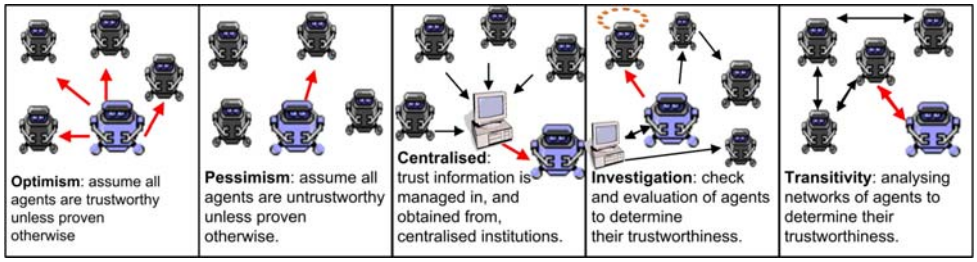

Fig.1. Five basic strategies from placing trust.

Much variation is possible within these strategies, and we do not claim that these are the only strategies available, or appropriate for the SW. Furthermore, it is possible to imagine systems that adopt a mixture of these strategies opportunistically, depending on what information is available to them at any moment, what risk they are prepared to bear, and what resources they have available to them. In other words, a system might change between strategies dynamically as circumstances (costs, risks, or the environment, for example) change. As an example, see [5], which switches between the first two of the strategies we outline, depending on its previous successes and failures with strangers. Different strategies get used as the system's understanding of the environment alters.

Note also that our discussion ignores the question of who takes the actual trust decisions. It is neutral on the question of how the preferences of people or policies of corporations interact with the agents' trust strategies. Clearly it is unlikely that software agents would get complete autonomy in important transactions.

3.1 Optimistic Systems:

Optimistic systems accept others unless there is reason not to trust. If the benefits of cooperation are relatively large or the costs of betrayal are relatively small, risk is low, and the gains from trust massively outweigh the gains from distrust. Such systems are also likely to benefit from decentralisation; self-policing ensures the shared ethics of the user population determine the policies of trust management [13].

Other systems exploit optimism for the purposes of a *bootstrapping process*. For example, in referral systems [27], each user has a personal agent which stores a user model. A user sends a query to his agent, which then suggests potential contacts to whom to send the query (which is an iterative process). In such a system users can form models of others on the basis of the replies they give. However, they need replies - and they need to act on them - if they are to gain any information that will enable them to adjust their acquaintance models. The system will bootstrap more quickly with an optimistic strategy.

Optimism Optimism is a very simple strategy. Basically it is the idea that an agent will trust another agent even if its performance is uncertain, unless there are positive reasons for not trusting it. The basic idea is that trust is the default attitude. Hence an agent will accept the bona fides of other agents offering services unless they fail one or more tests.

3.2 Pessimistic Systems:

CS AKTive Space (CAS) [23] is a SW application, which provides browsable representations of the discipline of Computer Science in the UK. The information is harvested off the

Web and stored by various techniques to acquire knowledge. The approach chosen by [23] is to rely on sites whose brands are trusted, for example the Web sites of computer science departments in the UK, thus increasing the change to getting sound data. [6] attempts to understand the SW as an ecology of agents, rather than a big database, and suggests the use of authorised delegations, that is the granting of a right to one agent by another, which must meet various constraints before they can be accepted by other agents. It is claimed that such systems, which take a generally pessimistic view of other agents and knowledge sources, have clear advantages in the distribution of trusting decisions ([12], [4], and section 3.3 below). [18] describe a pessimistic approach, though intriguingly hybrid. They suggest taking in information from past interactions about both the confidence of an agent in other agents, as well as their confidence in this agent. Given a quantity of such information, agents can be ranked in terms of their reliability. Such systems in effect take a high rank as evidence of reason for trust; hence in such a system many trustworthy agents may fail to be trusted.

Pessimism Pessimistic strategies restrict interactions with agents unless there is a reason to trust them. Note that the pessimism corresponds to trust via personal acquaintance in the offline world, which is the basic model of trust (local trust, [17]). Such a model of trust is not often capable of supporting and underlying very complex societies [7].

3.3 Centralised Trust Systems:

A system which helps users annotate information sources is described in [8]. It provides them with a formalism for expressing agreement/disagreement, and the argumentative stance of the source. This is then used to measure a context-sensitive evaluation of the source. [15] proposes a centralised agent to measure the reputation of Web services by monitoring and collecting client feedback, and making this information available to other agents.

Relying on centralised institutions to measure trust takes the burden off the interactive agents when deciding which agents to trust. However, such systems raise the question of how trustworthy are the sources of their trust information in the first place, and why such trust warehouses should be trusted at all [4]. One observation made in [20] with respect to *eBay* is that users feedback is almost always positive. The authors note that most people do not like giving negative feedback, unless revenge is a motivation. Clearly the centralised trust system, be it eBay or otherwise, is only as good as the information it is receiving and brokering; if, as with eBay, there is a bias in the information coming into the system and being displayed, then the strategy of other agents of shifting trust to the centralised system may well be non-optimal. Similarly, [6] argue against centralised units for measuring trust because of their scalability limitations and the implicit trust measurement mechanisms they adopt.

Centralisation Centralising trust involves laying off the costs of interacting with and investigating agents to a central institution or authority. The institution may certify a particular agent as willing and able to act in some specified way. If the agent bears a certificate, then it could be trusted. However, this does not obviate the need for trust, but the trust requirements are reduced. The agent now only needs to trust the institution - can it investigate potential agents thoroughly, and does it have the powers to sanction betrayals effectively? Some institution, like [8]'s TRELLIS system, merely holds the relevant collected information for users to process themselves.

3.4 Trust Investigation Systems:

In [24], a Bayesian system is describe which models trust in a P2P file distribution network. On such a network, peers make recommendations to each other about where suitable files might be found. The agents 'gossip' with each other, by exchanging and comparing their Bayesian networks. After this comparison, the agents update their trust ratings of each other, depending on whether they share similar preferences, on the assumption that an agent with similar preferences is more likely to give suitable recommendations than others.

In other words, the agents perform an investigation of the others in order to determine how likely it is that their recommendations will be useful. The agents do not simply receive the recommendations passively; they compare Bayesian networks, and undertake extra computational tasks, in order to determine their suitability.

Another example of this sort of approach is provided by systems that negotiate automatically to extract trust credentials from other parties. For example, the TrustBuilder system [25] works by iterative disclosure of credentials by the negotiating parties, in response to a series of requests for credentials, until either the set of credentials is exhausted, and the negotiation concludes unsuccessfully, or until sufficient trust between the parties has been established. The credentials are usually supplied by centralised authorities, and so this version of investigation has strong links with the previous strategy. But the point of a negotiation is that the agents themselves do (potentially a lot of) computation on the credentials, the context and the current state of negotiation to work out (a) which credentials to ask for, (b) how to evaluate responses, and (c) how much information to release at each stage in the negotiation.

Investigation This suggests a fourth strategy. Trust is a response to uncertainty. But trust imposes risks. Hence, to avoid some risk, one strategy is to reduce uncertainty by investigating or evaluating other agents to determine some salient details of operation. It is not passive; it actively tries to discover aspects of the environment that are relevant to reduce uncertainty.

3.5 Transitive Trust Systems:

In [10], authors argue that a likely future scenario will be a networked infrastructure of entities, linked by various ad hoc networking technologies. They use the small world theory [16], which hypothesises that any pair of objects in a random network will be connected by a relatively short chain of random acquaintances. This entails that if such mutual chains of acquaintances are used to determine initial trust between a pair of entities, then the method will scale up well because these chains are likely to be small.

Social network analysis techniques are used in [9] to measure trust over a Friend of a Friend (FOAF)[1] network, extended with trust relations. A similar approach of exploring webs of trust is described in [21], where users provide trust values for a number of other users, which are used to measure trust. These algorithms, as in [8], rely on manually-set trust opinions, which once more shifts the problem of trust to the source of the opinions.

As [3] are correct to argue, trust is not strictly transitive. If A trusts B, and B trusts (and maybe recommends) C, nothing follows about whether A trusts C. However, there are some circumstances where transitivity happens, at least locally. Similarly, transitive effects can be seen where trust is given a recursive definition [4].

[1] http://www.foaf-project.org/

One weakness of much of the work above is the lack of sensitivity to *context*. Context is a basic feature of trust [4] and must therefore be considered when dealing with it. For example you may trust a colleague to write a good project proposal, but you might not trust him to fix your car. Associating trust measurement with a specific context will inevitably increase complexity, but nevertheless is crucial for deriving meaningful trust values. Some notion of context could be obtained from the type of relations between networked agents, which plays a crucial role when measuring agent's reputation [22].

Exploring Transitivity These systems can be described as using the strategy of exploiting *transitivity*. The idea of this strategy is that an agent sends a message out about whether a potential agent is trustworthy. The network of acquaintances of that agent will then either send back an opinion based on experience, or pass the message onto its acquaintances, many of which will be unknown to the first agent. The aim is to increase the scope of an agent's knowledge by exploring the network feature of agents communities to bring in information from other, unknown, agents.

As noted above, the issue of context is important, and the trade-off between investigating context and relying on others' reports and recommendations shows the close relationship between the strategies of Investigation and Exploring Transitivity. In many actual systems and situations, there is likely to be a tight coupling between the two strategies, as investigation ameliorates some of the difficulties of exploiting transitivity.

4 Costs and Benefits

The main point of a trust strategy is to provide a way to operate under uncertainty, not taking too many risks, not missing too many opportunities, not deliberating too long before making commitments. Therefore, before setting up a plan of interaction between SW agents it is necessary to understand the risks and benefits associated with the selected trust strategy.

There are many different kinds of costs and benefits an agent might incur when communicating or dealing with other agents and services. Here we discuss four types of costs; *operational*, *opportunity*, *deficiency*, and *service charges*. We describe our estimations for each of these costs with respect to each of the five trust strategies discussed earlier, and summarise them in figure 2.

4.1 Operational cost

Operational costs are the expenses of operating a trust strategy. In other words, this is the cost of setting up and operating the whole trust plan. Therefore, the more complex the strategy is the higher the cost is expected to be.

The cost of operating an optimistic strategy is relatively small. Obviously it gets smaller the more optimistic the agent becomes, with a limiting case of zero when the agent takes all agents' bona fides as genuine. Pessimistic agents tend to run a set of tests to limit the number of agents they interact with. However, this does not necessarily lead to high operational costs. The one property that pessimism shares with optimism is that the strategy can be based on a very small and non-complex set of tests, which may involve merely the checking of identities. As the judgments about cooperation become more fine-grained, as for example with the investigation strategy, then operational costs will increase accordingly. One effect of trust is to cut out transaction costs, and naturally the effect of deciding to investigate rather than to trust is to reinstate them.

Agents in a centralised systems can look forward to very low operational costs because the complex parts of the strategy (e.g. investigations) will be handled by the institution itself (which can garner economies of scale). If the agent is part of a network where transitivity is adopted as a trust strategy, then it will incur some operational costs to implement and tune the techniques required to perform the network analysis. It may also incur costs as a by-product of being in a network, by having to pass on its experiences to other agents.

4.2 Opportunity cost

This is the cost of missing some possibility of generating benefit via interaction. The optimistic approach minimises opportunity costs. The more optimistic the agent, the less likely it is to miss some opportunity for interaction. On the other hand, there is the potential for a large opportunity cost with pessimistic agents by missing out on the potential to add value because of a reluctance to trust agents to perform necessary subtasks.

Opportunity costs may be higher with the centralised strategy than under the optimistic strategy as the agents' interaction will be limited to those certified by the institution. However, the cost should still be fairly low if many agents are certified. Opportunity costs are also likely to be high with the investigation strategy, unless the agent has the capacity and resources to investigate enough potential agents to keep opportunity costs down. With the transitivity strategy, the larger the network relative to the space of possible interactions, and the better its advice, the lower the opportunity costs.

4.3 Deficiency cost

Defecting agents fail to act as they claimed. The deficiency cost is the cost of betrayal by an agent. This cost is strongly associated with the amount of risk undertaken by an agent.

The risk of betrayal is high with optimistic agents, and so, the agent should plan for likely cost of an agent defecting. Equally for pessimistic agents the deficiency cost should be low. The main advantage of investigation is the lowering of risk by the simple expedient of lowering uncertainty. By performing an investigation or evaluation, the agent gains knowledge about whether and how the other agent is going to work.

The big advantage of a centralised system is that a small outlay of trust on the part of the agent still allows interaction with a large number of agents. However, this increases systemic risk. In a distributed system, a betrayal of trust will typically cause the agent to withdraw trust from the offending agent. But if an agent comes to another with a certificate issued by an institution, and then betrays that agent, the betrayed agent may well withdraw trust from that institution's procedures, and therefore refuse to cooperate with any of the institution's client base. The scalability problem of centralised systems adds to the systemic risk [4]. As the number of certified agents increases, then the degree of trust in the central authority may well begin to fall. Hence [4] recommend a strategy of pessimism, combined with a distributed system of trust management.

As for the transitivity strategy, if the network is large, then deficiency risk may also be large. But, unlike with an institution, it may be possible to weed out the source of bad advice from a network, and the network may be able to reconfigure itself and recover. Hence the advantage of exploring transitivity is that its costs may undercut those of the other strategies, and will also be spread, thereby hedging the risk.

4.4 Service charges

Agents may have to pay for purchasing services from other agents. Even on the most uncommercial views of the SW, there is little dispute that there will be a major effort required to populate it. Many people will of course be more than willing to post information on the SW, but the effort required to populate the space with enough RDF, and to ensure that the information posted will be that required or desired by other users will be non-trivial. Incentives of some kind will be required if the SW is to spread out of the academic ghetto.

Incentives and other charges may be high with the optimistic strategy for two reasons. First, the optimistic agent may well purchase more services from agents than those pursuing other strategies. Second, in the absence of strict monitoring of agent performance, higher payments may be required to ensure the alignment of interests.

Payment for services by pessimistic agents may be lower, as in some circumstances the agent can act on the understanding that its interests are aligned with those of the other agent. An additional advantage is that the smaller number of agents with whom the main agent interacts will reduce the complexity of any decision-making.

Adopting the investigation strategy may result in high operation costs, as its investigations themselves may well require consultation with many services. Similarly with the transitivity strategy - exploring the network may involve accessing fee-based services.

Operation costs of agents who rely on centralised units for handling trust is low. However, there might be a charge to be paid for this service. But because such institutions will probably serve many agents, they will benefit from economies of scale, and their charges should be much less than the accumulated costs of dealing with several distributed services.

Strategy / Cost	Optimism	Pessimism	Centralisation	Investigation	Transitivity
Operational cost	Low - does not require much policing.	Rises with complexity of filtering tests.	Low – the cost is embedded in the centralised service.	High – complex tests increase cost	Low if efficient methods are used.
Opportunity cost	Low - unlikely to miss many interactions.	High – does not check all possibilities.	High – decrease if many agents are certified.	High – only interact with investigated agents.	The larger the network, the lower the cost.
Risk	High - more risk when less cautious.	Low – won't interact with uncertain agents	Low – only legitimate agents are certified.	Risk is low by lowering uncertainty.	Rises with length of chains of referrals.
Deficiency cost	High - no check for malicious agents.	Low – malicious agents will be caught up front.	High – betrayal of one certified agent may affect the whole service.	Low –malicious agents banned if discovered	High - Behaviour of one agent might spread to network
Service payments	High – if max price/no. of interactions not set	Limited interactions results in limited charges.	Low – pay the centralised service only	High – may need to consult several services	May rise if when accessing fee-based services.

Fig. 2. Costs estimates for five trust strategies.

5 Meeting the Challenges of the SW

How do these strategies address the challenges for the SW set out in section 2? The cheap and simple strategies strategies; optimism and pessimism, meet most of the challenges half way, by avoiding most of the complexities of the situation (eg. binary modelling of trust makes its computation much simpler). Bootstrapping is trivial with optimism, but very challenging with pessimism. Both approaches however, fail to take account of the context sensitivity of trust judgements.

Investigation as a strategy can be very useful for bootstrapping trust, as some relevant properties of an agent may be known before it has actually begun to operate in a domain. Furthermore, investigation could meet the context challenge, assuming the methods of investigation were sensitive to the changes in context. However, the distributivity of the SW will make thorough investigation increasingly costly.

Centralisation is similar in some respects to investigation. If centralised authorities distribute certificates of trustworthiness in some domains that increase in complexity, it may be hard for such authorities to scale up. That does not mean that this strategy should not be used; only that such systems can only expect to flourish with a certain amount of flexibility. Users of competing authorities will need to decide between them, which may impose transaction costs.

The most interesting systems are those that exploit transitivity, in that they allow relatively flexible responses. For instance, a system using the transitivity strategy could cope with the problems of scale caused by increasing distributivity by pruning its searches; as trust is affected by the length of a chain of recommendations, falling as the chain gets longer, this pruning is unlikely to result in the loss of too much investigation. In this way, the analysis of a social network is analogous to the discovery of communities of practice [1], where shorter paths indicate stronger links.

Where most transitivity systems break down most seriously is on context dependence. Systems such as those of [9][21] tend to rely on fixed trust values explicitly given by users for others to use. This is of course valuable information, but rather than respecting the subjectivity of trust, it creates an objective sum of others' subjective valuations - not the same thing. One could imagine network analysis techniques, on the lines of [1], being exploited here. If the agent could specify which relationships in a representation of a network are the most important for it, in its specific context, then trust values could be determined dynamically depending on that specification.

6 Conclusions and Future Work

The SW, will be large, heterogeneous, dynamic and uncertain. Trust will inevitably be an issue. But for trust to flourish in such an environment, agents must be flexible, and adaptive to new contexts. To do this, it is important to reason at a strategic level, to think in terms of classes of acceptable outcomes. To this end, an understanding of how the costs and benefits of various strategies vary across contexts is essential. As we have seen, different strategies can be advantageous in different circumstances. In general, given the SW challenges, the strategy of exploring transitivity by trying to create chains of trusting judgements, seems quite promising, though in certain circumstances other strategies will reveal their strengths.

This paper has produced a preliminary classification of strategies based on an a brief survey of approaches. This classification certainly is not the final word. But when a robust classification is available, it will remain to produce quantitative evidence about which strategies are valuable where - and when this evidence is available, it might be possible for actual trust mechanisms to be provided by SW service brokers. Such evidence might well be produced on experimental testbeds such as that described in [11].

Acknowledgments. This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of the EPSRC or any other member of the AKT IRC. Thanks also to the anonymous referees for various helpful comments.

References

1. H. Alani, S. Dasmahapatra, K. O'Hara, and N. Shadbolt. Identifying Communities of Practice through Ontology Network Analysis. *IEEE Intelligent Systemss*,18(2): 18-25, 2003
2. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
3. T. Dimitrakos, J. Bicarregui. Towards modelling e-trust. *Proc. 3rd Panhellenic Logic Symposium*, Anogia, Greece, 2001.
4. L. Ding, L. Zhou, T. Finin. Trust based knowledge outsourcing for semantic web agents. In *Proc. IEEE/WIC Int. Conf. on Web Intelligence*, Beijing, China, 2003.
5. M. Feldman, K. Lai, I. Stoica, J. Chuang Robust incentive techniques for peer-to-peer networks In *Proc. 5th ACM Conference on Electronic Commerce*, New York, 2004.
6. T. Finin, A. Joshi. Agents, trust and information access on the semantic web. *SIGMOD Record*, 31, 2002.
7. F. Fukuyama. Trust: The Social Virtues and the Creation of Prosperity. NY: Free Press, 1995.
8. Y. Gil, V. Ratnakar. Trusting Information Sources One Citizen at a Time. *Proc. 1st Int. Semantic Web Conf. (ISWC)*, Sardinia, Italy, 2002.
9. J. Golbeck, B. Parsia, J. Hendler. Trust Networks on the Semantic Web. *Proc. of Cooperative Intelligent Agents*, Helsinki, Finland, 2003.
10. E. Gray, J. Seigneur, Y. Chen, and C. Jensen. Trust propagation in small world. In *Proc. 1st Int. Conf. on Trust Management (iTrust'03)*, 2003.
11. D. Huynh, N.R. Jennings, N.R. Shadbolt. Developing an Integrated Trust and Reputation Model for Open Multi-Agent Systems. In *Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multi-agent Systems (AAMAS)*, Columbia University in New York City, 2004.
12. L. Kagal, S. Cost, T. Finin, Y. Peng. A framework for distributed trust management. In *Proc. 2nd Workshop on Norms and Institutions in Multi Agent Systems*, Montreal, Canada, 2001.
13. S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proc. 12th Int. World Wide Web Conf.*, 2003.
14. B. Matthews and T. Dimitrakos. Deploying trust policies on the semantic web. In *Proc. 2nd Int. Conf. on Trust Management (iTrust'04)*, Oxford, UK, Mar. 2004.
15. E.M. Maximilien, M.P. Singh. An ontology for Web service ratings and reputations. *Workshop on Ontologies in Agent Systems, 2nd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Melbourne, Australia, 2003.
16. S. Milgram. The small world problem. *Psychology Today*, 61, 1967.
17. K. O'Hara. Trust: From Socrates to Spin. Icon Books, Cambridge, 2004.
18. S.D. Ramchurn, N.R. Jennings, C. Sierra, L. Godo. A computational trust model for multi-agent interactions based on confidence and reputation. In *Proc. 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, Melbourne, Australia, 2003.
19. S.D. Ramchurn, D. Huynh, N.R. Jennings. Trust in Multi-Agent Systems. *The Knowledge Engineering Review*, in press, 2004.
20. P. Resnick and R. Zeckhauser. Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. In M.R.Baye (editor), *Advances in Applied Microelectronics*, vol. 11, Elsevier Science, Amsterdam, 2002.
21. M. Richardson, R. Agrawal, P. Domingos. Trust Management for the Semantic Web. *Proc. 2nd Int. Semantic Web Conf.*, Sanibel Island, FL, 2003.
22. J. Sabater, C. Sierra. Reputation and social network analysis in multi-agent systems. *First Int. Conf. on Autonomous Agents and Multiagent system*, Bologna, pp:475-482, 2002.
23. N.R. Shadbolt, m. schraefel, N. Gibbins, S. Harris. CS AKTive Space: or how we stopped worrying and learned to love the semantic web. *Proc. 2nd Int. Semantic Web Conf.*, FL, 2003.
24. Y. Wang, J. Vassileva. Bayesian network-based trust model. In *Proc. of the 6th Int. Workshop on Trust, Privacy, Deception and Fraud in Agent Systems. 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems. (AAMAS)*, Melbourne, Australia, 2003.
25. M. Winslett, T. Yu, K.E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, L. Yu Negotiating trust on the web. In *IEEE Internet Computing*, 6(6), 30–37, Nov/Dec 2002
26. Y. Kalfoglou, H. Alani, M. Schorlemmer, C. Walton. On the Emergent Semantic Web and Overlooked Issues. In *Proc. of the Third Int. Semantic Web Con. (ISWC)*, Hiroshima, Japan, 2004.
27. B. Yu, M.P. Singh. Searching social networks. In *Proc. 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Sytems (AAMAS)*, Melbourne, Australia, 2003.

A Classification Scheme for Trust Functions in Reputation-Based Trust Management

Qing Zhang[1], Ting Yu[1], and Keith Irwin[1]

Department of Computer Science
North Carolina State University
{qzhang4, tyu, kirwin}@ncsu.edu

Abstract. Reputation is an important means to establish trust in decentralized environments such as the Semantic Web. In reputation-based trust management, an entity's reputation is usually built on feedback from those who have direct interactions with the entity. A trust function is used to infer one's trustworthiness based on such feedback. Many trust functions have been proposed in the literature. They are typically designed for specific application domains, thus differ in a variety of aspects, including trust inference methodologies, complexity and accuracy. In this paper, we propose a classification scheme for trust functions, which will help the systematic analysis and selection of trust functions for particular applications.

1 Introduction

The Semantic Web is visioned as the next Web. By instilling semantics into "flat" documents, the Semantic Web enables users to retrieve information that satisfies queries much more complex and expressive than today's keyword-based queries.

Due to the decentralized nature of the Semantic Web, an entity can publish whatever information without going through any scrutiny. Though seemingly relevant information can be retrieved, there is no assurance of its usefulness and correctness. Cryptographic techniques may be used to protect information confidentiality and integrity, but not its quality. To mitigate the problem, the trustworthiness of information sources needs to be considered.

Reputation is an important means to establish trust in decentralized environments. An entity's reputation is usually built on feedback from those who have direct interactions with the entity. Given a set of feedback, one's trustworthiness can be inferred through the use of *trust functions*. Trust functions have a direct impact on one's decision regarding information collection and other critical tasks, and is a key component to any reputation-based trust model. Many trust functions have been proposed, targeting at different application domains. They differ in a variety of aspects, including trust inference methodologies, complexity and accuracy. When a user or a group of users wants to join a decentralized system, a natural question is what trust function should be used for trust-related decisions. Instead of always designing their own trust functions from scratch, it will be beneficial if existing trust functions can be reused. To make a rational choice, it is necessary to have a classification scheme for trust functions so that their advantages and disadvantages for the problem at hand can be systematically analyzed.

Such a classification scheme is the focus of this paper. In section 2, we present a general framework for reputation-based trust. The framework not only includes relevant information for trust decisions, but also models the social interaction structure of a decentralized system. The framework thus can accommodate many of the existing trust models in the literature. Based on this framework, we propose a classification scheme for trust functions in section 3. The scheme has four dimensions, namely the nature of trust (subjective trust vs. objective trust), information collection (complete information vs. localized information), trust decisions (rank-based vs. threshold-based) and inputs of trust functions (opinion-based vs. transaction-based). We discuss the impact of each dimension on the properties of trust functions, and review some representative trust functions in the literature, based on the proposed classification scheme in section 4.

This work is not the first trying to classify trust functions. For example, Ziegler et al. [15] also provides a classification scheme of trust metrics. Some of their classification dimensions overlap with ours, while some others concern about other aspects of a trust model, such as where trust computation takes place. We will compare their scheme with ours when appropriate in this paper.

2 A Framework for Reputation-Based Trust

We assume that entities in a decentralized environment interact with each other through transactions. Transactions are not limited to monetary interactions. They also include activities such as retrieving information from a website, downloading files from a ftp server, etc. We further assume a transaction is uni-directional, i.e., given a transaction, there is a clear distinction between a service provider (server) and a service consumer (client). We introduce the following notion to facilitate our discussion.

Trustworthiness An entity's trustworthiness is an indicator of the quality of the entity's services. It is often used to predicate the future behavior of the entity. Intuitively, if an entity is trustworthy, it is likely that the entity will provide good services in future transactions. In most trust models [4, 6, 11], the domain of trustworthiness is assumed to be $[0, 1]$.

Feedback A piece of feedback is a statement issued by the client about the quality of a service provided by a server in a single transaction. In general, feedback may be multi-dimensional, reflecting the client's evaluation on a variety of aspects of a service, e.g., price, product quality and timeliness of delivery. For simplicity, we assume in this paper that feedback is one-dimensional, and is also from the domain $[0, 1]$.

Opinion An opinion is a user's general impression about a server. It is derived from its feedback on all the transactions that are conducted with the server. We also assume an opinion is one-dimensional and is from the domain $[0, 1]$.

In some trust models [3, 7], if an entity A has direct interactions with another entity B, then A's opinion of B is treated as B's trustworthiness from A's point of view. In some other models [4, 8, 11], A also needs to consider the information of B from other entities to infer the overall trustworthiness of B. Thus, in this paper we distinguish opinions from trustworthiness.

Source and Destination of Trust Evaluation If an entity A is interested in knowing the trustworthiness of another entity B, then we say A and B are the source and destination of a trust evaluation respectively.

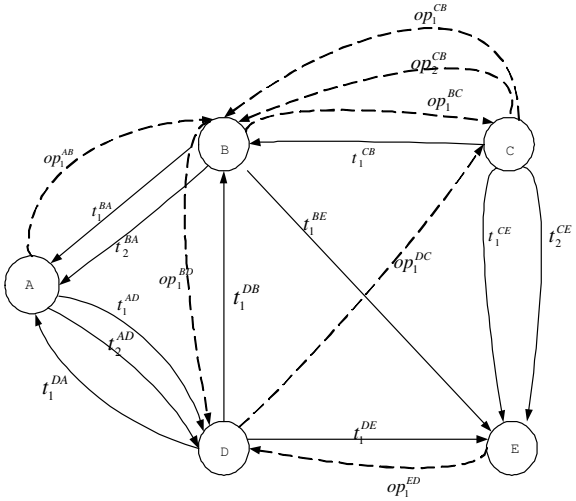

Fig. 1. An example trust graph

We model the relevant information for trust decisions as a directed multigraph $G(V, E)$, where V is a set of vertices and E is a set of labeled edges. V represents all the entities in an open system. They may provide service to others or request services from others, or both. We also call G a *trust graph* over V.

There are two types of edges: transaction edges and opinion edges. A transaction edge e from vertices A to B represent a transaction where B provides a service to A. The label of e contains the basic information of the transaction, e.g., the time of the transaction, the transaction type (e.g., file downloading and movie reviews), and the quantity of the services (e.g., total size of downloaded files and the number of ordered products). In particular, the label contains A's feedback on the transaction. There may be multiple transaction edges from A to B as more than one transaction may be conducted between them.

Similarly, an opinion edge from A to B represents A's opinion on B's service. There may also be multiple opinion edges from A to B, each of which represents A's opinion on a certain type of services provided by B. For example, A may have a good experience with B's reviews on movies but not those on small appliances. Figure 1 shows an example trust graph, where solid edges are transaction edges and dashed edges are opinion edges. Note that a trust graph may not be weakly connected since some users may not have any interactions with others, e.g., those just joining the system.

Let V be the set of entities in an open environment and $\mathcal{G}$ be the set of all the possible trust graphs over V. A *trust function* is a mapping $F : \mathcal{G} \times V \times V \rightarrow [0, 1]$. Intuitively, let A and B be the source and destination of a trust evaluation. Then $F(G, A, B)$ reports B's trustworthiness from A's point of view.

Many trust functions [7, 11, 3] assume that trust is transitive, i.e., if A trusts B and B trusts C, then A may also trust C. Thus transactions or opinions of the same type are used in trust functions. Some others [14] adopt a referral model, which makes a distinc-

tion between service recommenders and service providers. In other words, a good service provider may not offer useful information in recommending other service providers (e.g., due to competitions). Our framework accommodates both approaches since referrals can be modeled as a special type of transactions.

3 Trust Function Classification Scheme

Based on the above trust framework, we propose a classification scheme for trust functions. The scheme is composed of the following four dimensions.

3.1 Subjective Trust vs. Objective Trust

An entity's trustworthiness is often related to the quality of services it provides to others. If the quality of a service can be objectively measured, then an entity's trustworthiness for that service is called *objective trust*. For example, suppose a website provides specification information of automobiles. The quality (or accuracy) of such information can be indisputably checked against the official data released by manufacturers.

For some other services, their quality cannot be objectively measured. For example, given a movie review from a website, different people may have different opinions about its quality. It largely depends on each individual's taste and other subjective factors. In this situation, it is only meaningful to discuss the trustworthiness of the entity from a specific source's point of view. We call such trust *subjective trust*.

Intuitively, if the quality of a service can be objectively measured, then an entity's trustworthiness for that service reflects some intrinsic property of that entity, which should be independent of the source of the trust evaluation. For example, the accuracy of automobile specification information provided by a website should be the same to everybody. Formally, given a trust function F, if $F(G, A, C) = F(G, B, C)$ for any trust graph G, and any entities A, B and C, then we say F is suitable for objective trust evaluation, or F is an *objective trust function*.

An entity's subjective trust, however, may vary greatly when different sources of trust evaluation are considered. Thus, given a trust function F and an entity C, if there exist a trust graph G and entities A and B, such that $F(G, A, C) \neq F(G, B, C)$, then we say F is suitable for subjective trust evaluation, or F is a *subjective trust function*.

In general, subjective trust functions and objective trust functions are not comparable. They are suitable for different types of trust applications.

This classification dimension is similar to the distinction between global trust and local trust proposed in [15].

3.2 Transaction-Based vs. Opinion-Based

Some trust models rely on the information of individual transactions to infer an entity's trustworthiness, while others only request opinion information. To reflect the difference, we have the following definition. Given a trust graph $G(V, E)$, let $G_T = (V, E_T)$ where $E_T = \{e \mid e \in E \; and \; e \; is \; a \; transaction \; edge\}$. We call G_T the *transaction trust graph* of G, denoted. The *opinion trust graph* of G, denoted G_O, is similarly defined. Let F be

a trust function. If $F(G, A, B) = F(G_T, A, B)$ for all G, A and B, then we say F is a *transaction-based* trust function. Similarly, if $F(G, A, B) = F(G_O, A, B)$ for all G, A and B, then F is *opinion-based*.

Note that a transaction-based trust function does not always require detailed information of every transaction. Instead, some may only rely on some statistic information, e.g., total number of positive/negative transactions and the number of transactions during a certain period of time. But in general, it requires more information than opinion-based trust functions, inflicting a higher communication cost.

Since opinion-trust functions give each entity the autonomy to form their own opinions and conceal detailed transaction information, they are more privacy-friendly. Due to the same reason, however, opinion-based trust functions may be more easily influenced by malicious users. For example, Alice may have had a lot of transactions with Cathy, and forms an opinion on Cathy reasonably by using the percentage of positive transactions among all her transactions with Cathy. Another entity Bob, in the extreme case, may have an opinion on Cathy of value 0, even though he has never interacted with Cathy. By simply looking at these two opinions, it is hard to tell which opinion is more valuable for one's trust decisions[1].

3.3 Complete Information vs. Localized Information

Trust functions can also be classified according to the way information is collected. Some trust functions [12, 8] assume that every entity has the same access to all the transaction or opinion information. In other words, to apply a trust function, a complete transaction or opinion graph is a must. We call such trust functions *global trust functions*.

Another approach is to adopt a localized search process. Typically, it is assumed that an entity has several neighbors, who may or may not have interactions with the entity before. If Alice wants to evaluate Bob's trustworthiness, she will broadcast to her neighbors the requests for Bob's transaction/opinion information. This process continues until her neighbors have returned sufficient information for Alice to make a trust decision. To achieve better performance, information collection is usually a controlled "flooding" process. Therefore, the trust function is applied on a subgraph of the complete trust graph. Since each entity chooses their neighbors freely, different trust evaluation sources may construct different subgraphs. Thus we call trust functions of this kind *localized trust functions*. Intuitively, for a localized trust function, each entity typically has access to different information. A localized trust function is thus also subjective [15].

In general, localized trust functions scale better and are more suitable for decentralized environments. They also avoid the privacy concerns which may arise with the use of global trust functions. However, global trust functions tend to produce better results due to its access to a complete trust graph.

[1] Of course, a malicious users may issue biased feedback. But feedback has to be associated with a transaction trail, which can be signed by both the server and the client [9], or created by a trusted third party, e.g., in ebay. Thus, it is harder to influence a transaction-based trust function than an opinion-based trust function.

3.4 Rank-Based vs. Threshold-Based

Once a trust function returns the trustworthiness of a server, should we request services from that server? In other words, how should we utilize one's trustworthiness to make a trust decision? This question, in fact, relies on the nature of the output of a trust function.

For most trust functions, its returned trustworthiness can be interpreted as an approximation of some of the properties of a server. For example, if the trustworthiness of an automobile website is 0.8, we may think that approximately 80% of the information provided by the website is accurate. For such trust functions, it is appropriate to pre-define a threshold of trustworthiness to make trust decisions. For example, if a website's trustworthiness is over 0.9, then we trust information from that website. Thus, we call such functions *threshold-based*.

In some other trust functions, the calculated trustworthiness of a single entity alone does not convey much information. It becomes meaningful only when it is compared with the trustworthiness of other entities. In some sense, such trust functions return the relative ranking of an entity. We call such functions *rank-based*.

These two kinds of trust functions are suitable for different application requirements. If we would like to have certain quality assurance, then threshold-based trust functions are ideal. For rank-based trust functions, even if a server has a high ranking, it does not necessary mean that its service is of high quality. On the other hand, if we would like to know whether the quality of a server is among the top 10% of all service providers, then rank-based trust functions is more appropriate. Of course, we can also obtain the ranking information if we use a threshold-based trust function to infer every entity's trustworthiness. But it would be very expensive for large-scale decentralized systems like the Semantic Web.

The above four dimensions provide for easy matching between problems and trust functions. For each of these categories, it should be clear from the situation being addressed what sort of a trust function is needed. Questions about whether the services being measured are objective or subjective, whether every transaction or just overall opinion are important, whether complete information is available or local information should be used, and whether we care about an absolute threshold or a relative rank should be fairly easy to answer in most situations. We can, therefore, use these categories to conveniently identify which trust functions would be applicable for a given situation, significantly narrowing the process of choosing a trust function.

4 Classification of Existing Trust Functions

With the trust function classification scheme at hand, we now classify some existing works. A summary of the classification is given in table 1.

4.1 NICE

Lee et al. [7] proposed a trust inference scheme for NICE, a platform for Internet cooperative applications. In their scheme, after each transaction, the client A signs a cookie stating

functions	subj./obj.	trans./opinion	complete/localized	rank/thresh.
NICE	subj.	trans.	localized	thresh.
Evidence-based model	subj.	opinion	localized	thresh.
PeerTrust	obj.	trans.	complete	thresh.
EigenTrust	obj.	trans.	complete	rank
Reputation Inference	subj.	opinion	localized	thresh.
Trust for the Semantic Web	subj.	opinion	localized	thresh.
Heuristics Complaint Checking	obj.	trans.	complete	rank

Table 1. Classification of trust functions

the quality of the transaction to the server B, which later can be used by B to prove its trustworthiness to others, e.g., C. If B does not have cookies issued by C directly, it may consult its neighbors to collect chains of cookies from C to B. Such a set of chains of cookies indeed forms a subgraph G'_T of the transaction trust graph G_T. The final trustworthiness of B from C's point of view is obtained by calculating either the strongest path or the weighted sum of strongest disjoint paths in G'_T between C and B.

NICE is designed to help entities form cooperative groups. For the same service provider, entities from different groups may have different evaluations of its service. So the proposed scheme is a subjective trust function. Clearly, the scheme is transaction-based as cookies are collected before a subgraph of G_T can be formed.

4.2 Evidence-Based Model

Yu and Singh [14] proposed a distributed reputation management model which views feedback as evidence of the trustworthiness of a service provider. Trust evaluation is thus modeled as an evidence collection process.

In [14], a feedback can be either positive, negative or neutral. Given a set of feedback from user A to B, A's opinion on B is modeled as a triple $(m(T), m(T, -T), m(-T))$, where $m(T)$, $m(T, -T)$ and $m(-T)$ are the percentages of positive, neutral and negative transactions respectively. If A does not have direct interactions with B, A has to search for B's evidence through the help of her neighbors. This is where a subgraph of the trust graph G is formed. Then A will collect opinions from those who have direct interaction with B and form her own judgement of B's trustworthiness.

Similar to NICE, the evidence-based model is suitable for subjective trust evaluation. But it is opinion-based. No detailed transaction information is exchanged between entities.

4.3 PeerTrust

Xiong and Liu [12, 13] proposed PeerTrust, a reputation trust model for peer-to-peer systems. Using the P-Grid [1] data storage structure, PeerTrust assumes that every user is able to retrieve all the transaction information in a system. User i's trustworthiness is the normalized number of satisfactory services over the total number of transactions in which the user has taken part. It is further weighted by the trustworthiness of the feedback issuers and the quantity of each transaction.

Since all the users can get the complete information about any other user's transactions, they will end up with a common view of the trustworthiness of any user. PeerTrust is ideal for objective trust evaluation.

4.4 EigenRep

EigenRep [4] is a rank-based trust function. In EigenRep, system-wide complete transaction information is obtained via the CAN [10] data structure. The number of satisfactory and unsatisfactory transactions between each pair of entities is collected and used to construct a matrix. One special property of the matrix is that the entries in each row will add up to 1. The matrix will be repetitively multiplied with an initial vector, until it converges. The initial vector is a pre-defined system parameter which contains the default trustworthiness of each user. Each entry of the converged trust vector represents a user's final trustworthiness. Every user will get the same trust vector, since the matrix and the computation process is the same for all users. So this model would be a good candidate for objective trust evaluation.

Kamvar et al. [4] showed that the final trustworthiness of all users will also adds up to 1. Thus, only given the trustworthiness t_i of a particular user, it does not have any indication of the quality of that user's service. The output trust value just shows the comparative relations between users,i.e., given the global trust vector, we only can tell whether user i is more trustworthy than user j.

4.5 The Reputation Inference Algorithm

Compared with other trust functions, Golbeck and Hendler [3] propose a totally localized approach. In their model, each entity has several trusted neighbors. To infer the trustworthiness of a user B, A only polls her neighbors about their trust on B. This process continues recursively until it reaches entities who have interactions with B so that they can directly evaluate B's trustworthiness. The trustworthiness returned by each entity is either 1 (trusted) or 0 (untrusted). Once A receives all her neighbor's evaluation of B's trust worthiness, it simply takes a vote and decide whether it should trust B.

The trust graph is implicitly explored through recursive trust evaluation, which offers a simple protocol. On the other hand, since an entity does not have a relatively global view of the system and no transaction information is ever collected, it is critical to choose trusted neighbors. If one or more neighbors become malicious, an entity's trust decision can be greatly influenced.

4.6 The Trust Interpretation Model for the Semantics Web

Richardson et al. [11] discussed reputation-based trust management for the Semantic Web. The problem they considered is as follows. Suppose every user has local opinions on the rest of users in a system, which can form an opinion vector. How to get a global trust matrix such that each entry t_{ij} specifies user j's trustworthiness from user i's point of view, when considering other users' opinion. Two approaches are proposed based on different modeling of the problem.

The first approach assumes that t_{ij} in the global trust matrix only depends on the paths from i to j in the opinion trust graph G_O. Two auxiliary functions are defined to derive the global trust matrix from users' local opinions. Given a path from i to j, a concatenation function calculates i's indirect opinion on j along the path. An aggregation function calculates i's aggregated opinion on j over a set of paths from i to j. The global trust matrix can be derived through a sequence of matrix multiplication from users' local opinion vectors, by using the above two functions.

In the second approach, the same global trust matrix is built as in the first approach. But the problem is modeled as a random walk on a Markov chain, similar to the Page Ranking model in search engines [5]. t_{ij} in the matrix is interpreted as the probability that user i's surfer arrives at user j. Further, the second approach introduce a parameter λ to weight users' local opinion and the global trust matrix. Thus it allows users to maintain their own opinions which can be factored into their final trust decisions.

Though the trust function itself relies on complete information of the opinion trust graph, Richardson et al. designed an algorithm that only requires iterative information exchange between adjacent users in the trust graph. Further, this approach is opinion-based, which further reduces information exchange. Therefore, the global trust matrix can be derived efficiently.

4.7 Heuristics Complaint Checking

Aberer and Despotovic [2] considered the trust inference problem when users only issue negative feedback. Similar to PeerTrust, their approach also utilizes the P-Grid storage structure so that complaints issued by any entity can be retrieved. The trustworthiness of a user i can be calculated directly, based on the user's transaction history and compliants it receives. Aberer and Despotovic proposed a trust decision inequality to determine whether an entity should be trusted or not. The inequality does not provide a parameter explicitly for user customization. As long as two entities are both trusted, it cannot tell which one is more trustworthy. It can be viewed as a very course-grained rank-based trust function. Also, since every entity has access to the same information and uses the same trust function, they will always reach the same trust decision on a given user.

5 Discussion

Credentials are another important means to establish trust in decentralized systems. Credential-based trust and reputation-based trust are complementary to each other. On one hand, credential-based trust policies help avoid a large class of unreliable or irrelevant entities for a particular application, and make reputation collection more accurate and efficient. This is especially desirable for large-scale open systems. On the other hand, reputation-based trust is particularly useful when some of the authorities along a credential chain cannot be unconditionally trusted and continual history-based trust evaluation is required. It is desirable to design a comprehensive trust model which combines the strength of both. One key issue is to design a policy model that seamlessly integrates constraints on both credentials and reputations into trust policies.

6 Conclusions and Future Work

Trust functions are a central component to reputation-based trust management. An appropriate classification scheme of trust functions will help us systematically analyze their properties and choose the right one for a particular application. In this paper, we propose such a classification scheme based on a generic trust framework. We further review some representative trust functions in the literature according to the classification scheme.

As a part of our future work, we would like to investigate the application of reputation-based trust models in wireless network routings and distributed system load balancing.

References

1. K. Aberer. P-Grid: A self-organizing access structure for p2p information systems. In *Cooperative Information Systems, 9th International Conference (CoopIS)*, 2001.
2. K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM)*, 2001.
3. J. Golbeck and J. Hendler. Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-based Social Networks. In *International Conference on Knowledge Engineering and knowledge Management (EKAW)*, Northamptonshire, UK, Oct. 2004.
4. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. EigenRep: Reputation Management in P2P Networks. In *World-Wide Web Conference*, 2003.
5. R. Lawrence, B. Sergey, M. Rajeev, and W. Terry. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Department of Computer Science, Stanford University, 1998.
6. S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative Peer Groups in NICE. In *INFOCOM*, 2003.
7. S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative Peer Groups in NICE. In *IEEE Infocom*, San Francisco, CA, Apr. 2003.
8. L. Mui, M. Mohtashemi, and A. Halberstadt. A Computational Model of Trust and Reputation. In *35th Hawaii International Conference on System Science*, 2002.
9. J. Peha. Making Electornic Transactions Auditable and Private. In *Internet Society (ISOC) INET*, San Jose, CA, June 1999.
10. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, 2001.
11. M. Richardson, R. Agrawal, and P. Domingos. Trust Management for the Semantic Web. In *Proceedings of the Second International Semantic Web Conference*, 2003.
12. L. Xiong and L. Liu. Building Trust in Decentralized Peer-to-Peer Electronic Communities. In *The 5th International Conference on Electronic Commerce Research. (ICECR)*, 2002.
13. L. Xiong and L. Liu. A reputation based trust model for peer-to-peer ecommerce communities. In *IEEE International Conference on E-Commerce (CEC)*, 2003.
14. B. Yu and M. P. Singh. An Evidential Model of Distributed Reputation Management. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2002.
15. C.-N. Ziegler and G. Lausen. Spreading Activation Models for Trust Propagation. In *IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE '04)*, 2004.

An Initial Investigation into Querying an Untrustworthy and Inconsistent Web

Yuanbo Guo and Jeff Heflin

Dept. of Computer Science and Engineering, Lehigh University, Bethlehem, PA18015, USA
{yug2, heflin}@cse.lehigh.edu

Abstract. The Semantic Web is bound to be untrustworthy and inconsistent. In this paper, we present an initial approach for obtaining useful information in such an environment. In particular, we replace the question of whether an assertion is entailed by the entire Semantic Web with two other queries. The first asks if a statement is entailed if a set of documents is trusted. The second asks for the document sets that entail a specific statement. We propose a mechanism which leverages research on assumption-based truth maintenance systems to efficiently compute and represent the contexts of the statements and manage inconsistency. For these queries, our approach provides significant improvement over the naïve solution to the problem.

1 Introduction

Since the Semantic Web is intended to mirror the World Wide Web, it will be produced by numerous information providers with different levels of credibility, and will be used by information consumers who have different opinions on who or what is trustworthy. Some researchers have investigated methods for computing who a user should trust in such environments. In this work, we take a different approach: we investigate how to build a Semantic Web search engine that can tell the user what sources support each answer to a query so that the user could decide if they trust those sources. In addition, we aim at a system capable of efficiently answering queries once the user has decided what sources they trust and when they change their mind.

We will assume a document collection D consisting of N OWL Lite [3] documents, labeled as D_1 to D_N. We also assume that this collection can be harvested from the Internet at a rate such that the information maintained by a webcrawler is current enough to be of value. Note, our focus on a centralized search-engine approach is based on its success in the contemporary Web and on the fact that much research needs to be done before distributed queries can reach a comparable response time. Finally, we will assume that users are primarily interested in extensional queries, and will focus on queries about the instances of a class. We denote by a:C an assertion that individual a is an instance of class C.

Before we can formally define our problem, we must introduce two definitions. First, a set of documents D entails a statement φ iff φ is entailed by the union of the imports closure [4] of every document in D. As such, it is possible that a pair of documents might entail something that is not entailed by either document alone. Sec-

ond, a set $D_{sub} \subseteq D$ is a minimal consistent subset of D that entails φ iff D_{sub} is consistent, and D_{sub} entails φ and there is no subset of it that entails φ. Note, for a given φ, there may be multiple such sets.

Based on this, we propose two kinds of queries to be answered by the system:

- Q1: Given a trusted subset D_{sub} (of size M)[1], is D_{sub} consistent and does it entail an assertion a:C?
- Q2: What are the minimal consistent subsets of D that entail an assertion a:C?

We also take into account inconsistency in the queries. In classical logic, everything can be deduced from an inconsistent knowledge base. However, in many Semantic Web applications, this is not desirable and it is crucial to be able to identify inconsistent document sets in order to avoid inappropriate use of the semantic data. By these queries, we suggest that inconsistency can be managed, not by changing the underlying logic, but by changing the kind of queries we pose to the Semantic Web.

The rest of the paper is organized as follows. Section 2 looks at the naïve approach to answer the above two queries. Section 3 describes in detail an improved approach. Section 4 discusses related work. Section 5 concludes.

2 A Naïve Approach

1) Answering Q1

To answer Q1, we first combine the documents in D_{sub}. This involves loading them into a single knowledge base. Then if the knowledge base is found inconsistent, the answer to Q1 is false; otherwise, we query about a:C on the knowledge base. The result is then the answer to Q1. Such a query can be executed using a description logic reasoner that supports realization such as Racer [6].

2) Answering Q2

To answer Q2, we repeat Q1 against each applicable subset of D. We enumerate the subsets of D in increasing order of their sizes. In order to ensure that only minimal consistent subsets are returned, we keep track of those subsets that either answer yes to Q1 or are inconsistent so that we could skip all the supersets of them later on. This works because OWL is monotonic. The answer to Q2 will then be the document sets which have answered positively to Q1 during the test.

Next we analyze the complexity of this approach. To facilitate the analysis, we assume that the average size of the documents in D is S_D, and so is the average size of the documents in $D_{sub.}$

1) Answering Q1

It is known that reasoning on a language like OWL Lite, which maps to SHIQ(D+), is expensive with worst case N_{exp}Time complexity. Therefore, we could expect that the dominant factor of the complexity of answering Q1 is the time spent on reasoning including consistency check and instance query. We denote it by $T_{inf}(M*S_D)$. For simplification, in the subsequent discussion, we will not make distinction between different sorts of reasoning processes, instead, we generally refer to their time complexity as

[1] This set might be explicitly specified by the user or be determined by some certification authority that the user has specified.

$T_{inf}(s)$ wherein s is the size of the knowledge base measured by the total size of the documents loaded into it.

2) Answering Q2

Suppose k is the total number of subsets that have been tested with Q1, then the best case occurs when every document in D either entails the target assertion or is inconsistent. In that case, k equals to N and the time complexity is $N*T_{Q1}(S_D)$, wherein $T_{Q1}(s)$ stands for the time complexity of answering Q1 on a document set of size s. On the contrary, the worst case happens when none of the subsets could be skipped by the strategy and we are forced to do the query on all of them. In that case, k is as large as 2^N-1, and the time complexity is $O(2^N)* T_{Q1}(N*S_D)$.

This approach has several drawbacks. First, it is incapable of reusing the results of expensive inference from the preceding queries. For instance, if a Q1 query is repeated, we have to carry out the same process all over again. Answering Q2 is similar in this aspect. Second, the scalability of this approach is a problem especially for answering Q2. Again, the complexity cannot be amortized over multiple queries.

3 An Improved Approach

3.1 Assumption-Based Truth Maintenance System

Our approach builds on the concept of assumption-based truth maintenance system (ATMS) [1]. Like the conventional justification-based truth maintenance system (JTMS) [2], ATMS makes a clear division between the problem solver and the TMS, as shown in Fig. 1. The TMS functions as a cache for all the inference made by the problem solver. Thus inferences, once made, need not be repeated, and contradictions, once discovered, are avoided in the future. Unlike JTMS which is based on manipulating justifications, ATMS is, in addition, based on manipulating assumption sets. In an ATMS, every datum is labeled with the sets of assumptions, a.k.a. environments, under which they hold. These assumption sets are computed by the ATMS from the problem solver supplied justifications. Consequently, ATMS makes it possible to directly refer to a context, defined as a given environment plus all the data derivable from it. Thus context switching, which is very expensive in JTMS, is free in ATMS. Therefore, ATMS can work more efficiently than JTMS for problem solving by exploring multiple contexts simultaneously.

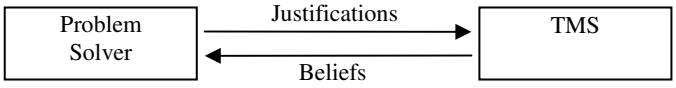

Fig. 1. ATMS Components

3.2 Document Preprocessing

Generally speaking, our approach aims at improving the scalability and efficiency by reusing the results of document processing, especially reasoning. This is realized by adding to the document processing a new functionality of figuring out and recording the

context of each encountered statement. Here we define: a "context" of a statement is a minimal consistent document set that entails the statement. This definition is sufficient because OWL is monotonic: if a statement is entailed by a set of documents, it is also entailed by any superset of that set; likewise, if a document set is inconsistent, each of its supersets will be inconsistent too.

We adopt ATMS to streamline the management of such contexts. In an ordinary ATMS, each node is associated with a proposition and a justification is a record of logical inference made between those propositions. In our approach, we use ATMS in an unconventional way. We use the ATMS nodes to represent two types of objects. We use an assumption node to represent a single document from D. We call the node a document node. And we use a derived node to represent a set of documents, in other words, the combination of these documents. We call the node a combination node. Following the notation in [1], we use Γ_d to denote a document node representing document d, and γ_v a combination node representing document set v. A justification $\Gamma_{d1},\ldots,\Gamma_{dn}=>\gamma_v$ is then interpreted as: the conjunction of the statements entailed by documents $d_1,\ldots,d_n$ implies the statements entailed by the document set represented by v. Moreover, a justification $\Gamma_{d1},\ldots,\Gamma_{dn}=>\perp$ conveys the information that document set $\{d_1,\ldots,d_n\}$ is inconsistent. It can be interpreted in a similar way when the antecedents of the justification contain combination nodes. Fig. 2 is an example ATMS for four documents, among which {D1, D2}, {D1, D3} and {D3, D4} are inconsistent. We will introduce the algorithm for constructing such ATMS at the end of the section.

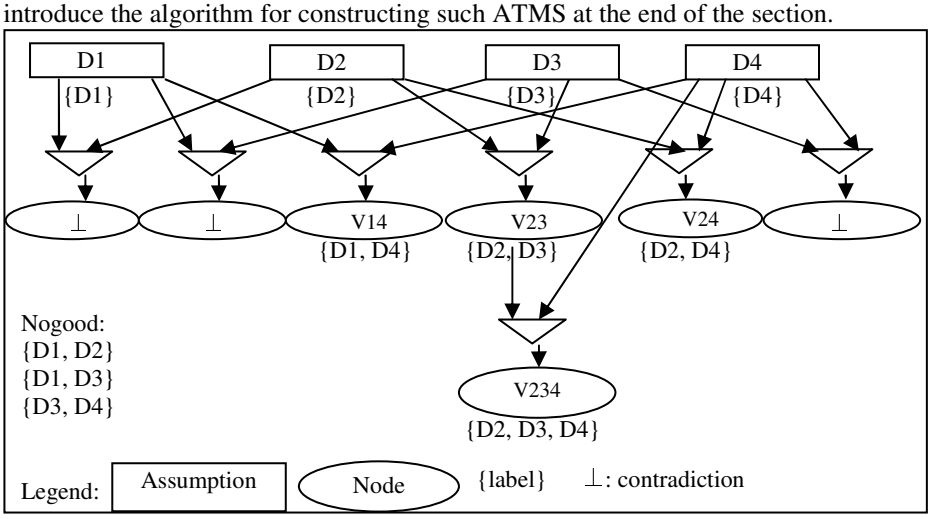

Fig. 2. An example ATMS network

There are several reasons for us to associate an ATMS node with a document or a document set as opposed to a statement. First is the scalability consideration. The scale of data makes it impossible to represent each statement individually and to provide a specified justification for it. Second, we assume that documents are all or nothing, i.e., we trust either the whole content of a document or none of it. One other minor reason is that since our description logic reasoners are black boxes, we cannot easily determine exact justifications at the level of a statement. We instead must determine them at the document level. As a result, an ATMS node in our system essentially points to a set of

statements and it serves as the media of the context of those statements: the environment of such a node is just a minimal consistent document set which entails the statements associated with the node.

Now what we need to do is to store the statements together with their contexts. To make our system more scalable, we do not store the deductive closure of the knowledge base. We observe that once subsumption has been computed, a simple semantic network is sufficient for answering queries about the instances of classes. Therefore, we only store the subsumption relations which are not redundant (for example, C1⊆C3 is redundant given C1⊆C2 and C2⊆C3) and the most specific classes of each instance. However, to answer the queries presented in Section 1, we also need context information. As a result, what is stored can be seen as a semantic network whose links are "annotated" by the contexts, as depicted by Fig. 3. As we will show in next section, this allows us to replace the expensive description logic reasoning with a much simpler semantic network inference-like procedure during query answering. In this way, we find a balance between doing some precomputation at loading time in order to save query time while controlling storage requirements.

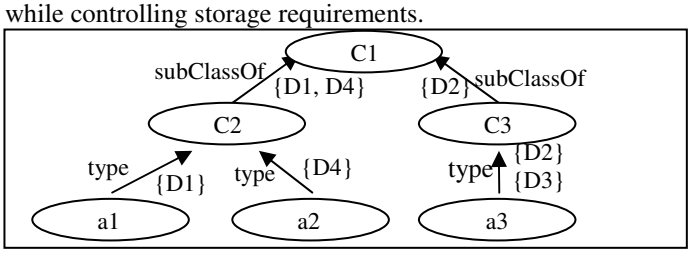

Fig. 3. An "Annotated" Semantic Network

Also for scalability, we store the "annotated" semantic network in a relational database. We use two kinds of tables. The first is a set of class instance tables, where there is one for each class. Each tuple of the table contains an individual of the class and a supporting node. By supporting nodes, we mean those nodes whose environment can entail that the corresponding individual is an instance of the class. Recall that a node's environment is a set of documents. Since the same concept assertion may hold in different document sets, an individual may have multiple supporting nodes with respect to a specific class. The second kind of table is a class taxonomy table, which records the class subsumption. Each tuple in the table consists of a superclass, its subclass, and a supporting node for the subsumption relation. Again, a subsumption may be supported by multiple nodes. Table 1 and Table 2 show what these tables look like for the semantic network displayed in Fig. 3.

Table 1. Class Instance Tables

Class C2

Individual	Supporting Node
a1	D1
a2	D4

Class C3

Individual	Supporting Node
a3	D2
a3	D3

Table 2. Class Taxonomy Table (*combination node of D1 and D4)

SuperClass	SubClass	Supporting Node
C1	C2	V14*
C1	C3	D2

Now we give the algorithm for processing a document (PROCESS-DOC). Due to space constraints, we intersperse the description within the pseudo code. The basic idea is, when a document is newly added, we apply inference on it and store the statements

entailed by it. Then we combine it with each of applicable subsets of the preceding documents and store the entailed statements by the combination.

```
1  procedure PROCESS-DOC(Dk)
2  {Assuming documents D1,…,Dk-1 have already been processed}
3  begin
4    LOAD(Dk);  /*load the document into a knowledge base*/
5    ADD-DOC(Dk);
6    if Dk is consistent then
7      for each non-empty subset s of {D1,…,Dk-1} do[2]
8        if ATMS: CHECK-NOGOOD(s)=false[3] then COMBINE-DOCS(s, Dk);
9  end
10 procedure ADD-DOC(Dk)
11 begin
12   ATMS: ADD-ASSUMPTION(Dk); /*add an assumption node for Dk*/
13   DO-INFERENCE(Dk);  /*apply DL inference on Dk*/
14   if Dk is inconsistent then ATMS:NOTIFY-JUSTIFICATION("ΓDk => ⊥");
15   else STORE-STATEMENTS(Dk, ΓDk);
16 end
17 procedure COMBINE-DOCS(set, Dk)
18 begin
19   set_node := the representative node of set;
20   Vnew := LOAD(set U {Dk}); /*load Dk and all documents in set
           into a knowledge base, Vnew being the combination*/
21   DO-INFERENCE(Vnew); /*apply DL inference on Vnew*/
22   if Vnew is inconsistent then
23     ATMS: NOTIFY-JUSTIFICATION("set_node, ΓDk  => ⊥");
24   else begin
25     ATMS: ADD-NODE(Vnew); /*add a new node representing Vnew*/
26     ATMS: NOTIFY-JUSTIFICATION("set_node, ΓDk => γvnew");
27     STORE-STATEMENTS(Vnew, γvnew);
28   end
29 end
30 procedure STORE-STATEMENTS(d, node)
31 {node is the ATMS node representing document set d;}
32 begin
33   for each non-redundant concept axiom C1⊆C2 in d do
34     ADD-TO-TAXONOMY-TABLE[4](C1⊆C2, node);
35   for each concept assertion a:C in d wherein C is the most
     specific class of a do
36     ADD-TO-INSTANCE-TABLE[5](a:C, node);
37 end
```

[2] We ignore it for brevity, but owl:imports can be taken into account for document combination. For instance, if one document imports another, we can skip the combination of both of them.

[3] One important functionality of ATMS is to record the assumption sets that have caused contradictions in a so-called nogood list. For instance, if we notify the ATMS that D_k is inconsistent, it will record $\{D_k\}$ as a nogood environment.

[4,5] Both procedures guarantee that a statement will ultimately be stored only with the nodes representing its contexts, i.e., the minimal consistent document sets that entail the statement.

3.3 Query Answering

Based on the above preprocessing, we can make the query answering more lightweight by reducing them to simple operations involving multiple table lookups. The algorithms are listed below. TEST-INSTANCE answers Q1 with individual *a*, class *C*, and a set of documents *set*. If *set* is inconsistent, we return false to the query (Line 4). Otherwise, we search for *a* in *C*'s table (Line 5, TEST-INSTANCE1). If we find a tuple of *a* such as its supporting node has an environment which is subset of *set*, we answer yes to the query (Line 24). If we could not directly find a matching tuple in the instance table, we will resort to the class taxonomy table (Lines 6-16). We search for the subclasses of *C* in *set*, and repeat the test with those subclasses (Line 12).

```
 1 procedure TEST-INSTANCE(a, C, set) return true or false
 2 begin
 3   {a: an individual; C: a class; set: a subset of D;}
 4   if ATMS: CHECK-NOGOOD(set)=true then return false;
 5   if TEST-INSTANCE1(a, C, set)=true then return true;
 6   else begin
 7     search the class taxonomy table for C;
 8     for each found tuple t do begin
 9       n := t.SupportingNode;
10       env := ATMS: GET-ENVIRONMENT(n);
11       if env ⊆ set then
12         if TEST-INSTANCE(a, t.SubClass, set)=true then
13          return true;
14     end
15     return false;
16   end
17 end
18 procedure TEST-INSTANCE1(a, C, set) return true or false
19 begin
20   search the instance table of C for a;
21   for each found tuple t do begin
22     n := t.SupportingNode;
23     env := GET-ENVIRONMENT(n);
24     if env ⊆ set then return true;
25   end
26   return false;
27 end
```

QUERY-INSTANCE answers Q2, also by consulting the information in the tables. But unlike TEST-INSTANCE, when two elements, one from the taxonomy table and the other from an instance table, are used simultaneously to derive an answer, the algorithm adds the union of the environments of their support nodes to the result (Line 20). In addition, it guarantees that what are finally returned are only those minimal environments, i.e., document sets. This is covered by INSERT-CONTEXT and INSERT-CONTEXTS in Lines 4, 9 and 20.

```
 1 procedure QUERY-INSTANCE(a, C) return a document set
 2 {a: an individual; C: a class; results := {};}
 3 begin
 4   INSERT-CONTEXTS(results, QUERY-INSTANCE1(a, C, {});
```

```
 5   search the class taxonomy table for C;
 6   for each found tuple t do begin
 7     n := t.SupportingNode;
 8     env := GET-ENVIRONMENT(n);
 9     INSERT-CONTEXTS(results,QUERY-INSTANCE(a,t.SubClass,env));
10   end
11   return results;
12 end
13 procedure QUERY-INSTANCE1(a, C, set) return a document set
14 {a: an individual; C: a class; results := {};}
15 begin
16   search the instance table of C for a;
17   for each found tuple t do begin
18     n := t.SupportingNode;
19     env := GET-ENVIRONMENT(n);
20     INSERT-CONTEXT(results, set U env);
21   end
22   return results;
23 end
```

3.4 Complexity Analysis

Now we analyze the computational complexity of our approach. As with the naïve approach, we focus on the most significant operations.

1) Answering Q1

Our approach has reduced query answering to table searching and eliminated the need of doing inference. How many database operations are required depends on how many document sets the statements considered in the process have as their contexts. In the best case, the number is at constant level. In the worst case, however, the number is $O(2^N)$, e.g., when the statement in the query is entailed by the maximum number of subsets of D such that no set contains another. Nevertheless, we could expect that in a real application, most of the statements will only have a handful of, if not one, document sets in D as their contexts. Therefore, the time complexity will be very close to the best case.

2) Answering Q2

As shown in QUERY-INSTANCE, answering Q2 has been realized in similar algorithm to that of Q1, except that the algorithm has to examine all possible contexts of a statement since no candidate is specified as in Q1. But this does not increase the order of complexity. In other words, we have achieved a complexity of Q2 similar to 1). This is a significant improvement compared to the naïve approach.

3) Document preprocessing

Our approach reduces query time by doing extra work when loading documents. There are three major kinds of work: ATMS related operations, database operations, and inference. The most significant task is inference. ADD-DOC does inference on documents while COMBINE-DOCS does inference on document sets. Therefore roughly, the time complexity of both procedures are $T_{inf}(N*S_D)$. Since we try to combine a newly added document with every subset constituted by its preceding document, there are potentially $O(2^N)$ such subsets, which means COMBINE-DOCS has to be

invoked for $O(2^N)$ times in the worst case. This results in a worst case complexity of $O(2^N) * T_{inf}(N*S_D)$.

However, this complexity can be alleviated in the case when a document set is identified inconsistent at some time and a significant number of combinations involving that set are avoided later on. The example in Fig. 2 demonstrates this. It is similar in the case when some documents import others. In addition, considering the improvement on query efficiency, we could argue that the complexity of the document processing in advance could be amortized over a large number of queries, since queries are significantly faster here than in the naïve approach.

4 Related Work

Trust systems for the Semantic Web as in [7, 8, 9, 10] are developed to compute whether to trust a Semantic Web resource depending on certain factors such as its source. Clearly our work is not about such kind of trust system. Our system deals with the untrustworthy Semantic Web from a perspective of extensional queries and leaves the determination of who to trust in the hands of the user. However, it is possible to integrate our system with other trust systems. For example, another system could determine the trusted set for Q1, or the results of our Q2 could be used as input into a system trying to determine some form of community based trust.

Sesame is another Semantic Web system that makes use of a truth maintenance system. In particular, it uses a simplified JTMS to track all the deductive dependencies between statements and to determine which other statements have to be removed as a consequence of a single deletion [5]. Since we are essentially dealing with multiple contexts, the ATMS is much more suitable for us. In addition, their work tries to find out the dependency between statements while ours deals with the justification on statements at document level.

Finally, much work has been done in the logic community to study paraconsistent logics that allow reasoning with inconsistent information. Examples of such logical systems include bilattice-based logics [11, 12] and annotated logics [13, 14, 15]. As noted at the beginning of the paper, our work does not aim at developing the underlying logic to handle inconsistency. Instead, we proposed to manage inconsistency by changing the kind of queries we pose to the Semantic Web.

5 Conclusions and Future Work

In this paper, we considered the issue of how to obtain useful information on the inherently untrustworthy and inconsistent Semantic Web. We proposed an approach to build a system that could answer two kinds of queries. One asks if a statement is entailed by a trusted set of documents and that set is consistent. The other asks for the minimal consistent document sets that entail a specific statement as a way to help the user to decide if they trust those sources. We employ an assumption-based truth maintenance system (ATMS) to efficiently represent document sets as the contexts of the statements entailed by them. Also we leverage the mechanism of ATMS for in-

consistency management. Based on that, we introduced a mechanism which preprocesses the documents and caches the complex inference together with statement context information, and answer the queries based on these. We showed how our approach greatly improves the efficiency with respect to the proposed queries. Another characteristic of our approach is that it seamlessly integrates the task of query answering and inconsistency management in one framework. In this paper, we have concentrated on the queries about concept assertions. However, the approach presented here can be easily extended to support role assertions, for example.

For future work, we will look into ways to further improve the scalability of our approach, especially to reduce the average cost in the preprocessing. One of our plans is to devise a mechanism that discovers in advance if nothing new can be entailed by a combination of documents and thus allows us to omit the combination. Also we intend to transfer the current approach into a distributed one, possibly based on the work on distributed ATMS like [16].

References

1. Kleer, J. de. An assumption-based TMS. Artificial Intelligence, 28(2), 1986.
2. Doyle, J. A truth maintenance system. Artificial Intelligence 12(1979).
3. Bechhofer, S. et al. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/
4. Patel-Schneider, P.F. ed. OWL Web Ontology Language Semantics and Abstract Syntax. http://www.w3.org/TR/owl-semantics/
5. Broekstra, J. and Kampman, A. Inferencing and Truth Maintenance in RDF Schema: exploring a naive practical approach. In Workshop on Practical and Scalable Semantic Systems (PSSS). 2003.
6. Haarslev, V. and Moller, R. Racer: A Core Inference Engine for the Semantic Web. In Workshop on Evaluation on Ontology-based Tools, ISWC2003.
7. Golbeck, J., Parsia, B., and Hendler, J.Trust networks on the semantic web. In Proc. of Cooperative Intelligent Agents. 2003.
8. Klyne, G. Framework for Security and Trust Standards. In SWAD-Europe. 2002.
9. Richardson, M., Agrawal, R., and Domingos, P. Trust Management for the Semantic Web. In Proc. of ISWC2003.
10. Gil, Y. and Ratnakar V. Trusting Information Sources One Citizen at a Time. In Proc. of ISWC2002.
11. Ginsberg, M.L. Multivalued logics: A uniform approach to inference in artificial intelligence. Computer Intelligence, 4(1988).
12. Fitting, M.C. Logic Programming on a Topological Bilattice. Fundamenta Informaticae, 11(1988).
13. Subrahmanian, V.S. On the Semantics of Quantitative Logic Programs . In IEEE Symposium on Logic Programming. 1987.
14. Blair, H.A. and Subrahmanian, V.S. Paraconsistent Logic Programming. Theoretical Computer Science, 68(1989).
15. Kifer, M. and Lozinskii, E.L. A logic for reasoning with inconsistency. Journal of Automated Reasoning, 9(2), 1992.
16. Malheiro, B., Jennings, N., and Oliveira, E. Belief Revision in Multi-Agent Systems. In Proc. of the 11th European Conference on Artificial Intelligence (ECAI'94). 1994.

A fuzzy model for context-dependent reputation

(Trust, Security and Reputation Workshop at ISWC 2004, Hiroshima, Japan.)

Victor S. Grishchenko

Ural State University, gritzko@ural.ru

Abstract. This paper extends the previous work on constructing reputation axiomatics [28] to define context-dependent reputation (e.g. occupation-specific). In short, to employ theoretic-set approach, both notions of reputation and recommendations are downcasted to the basic notion of responsibility. Notion of "a context" is understood as a synonym of "a universe of discourse".

1 Rationale

First, we have to understand the dimensionality of the problem. In the first approach we may expect the number of variables to be equal to the number of participants, n. Every variable represents a public reputation of some participant regarding some fixed topic. Numerous reputation services use this approach, such as e-mail blacklists and whitelists or extensively studied eBay[10] reputation system. There are reputation models which involve explicit or implicit recommendations to calculate n public reputation values, such as EigenTrust algorithm[17] and a family of algorithms based on Markov chains (e.g. PageRank[1]).

Still, there is no evidence that all participants share the same opinions or the same law. So, generally, reputation have to be recognized to be an opinion. This raises dimensionality of a perfect map of reputation to n^2. Straightforward aggregation is meaningless here: 1 billion of Chinese people think that something is good, but my relatives think it is bad. Opinion is meaningful in the context of the owner!

The task of n^2 data storage could hardly be solved with a central database or Distributed HashTable[21]. Considering context issues, it seems natural to let every participant host own opinion and experience, own "map of trust". At the same time we know that most participants can not support a map of size n (in other words, to trace every other participant). Many researchers considered an idea of Web-of-Trust, a graph formed by participant's expressed pairwise directed trust relationships [7, 13, 14, 16, 19, 20, 22, 27]. In this web a trust between distant entities is derived as a function of chains (paths) connecting those entities. M. Richardson et al [15] provide a good formal description of the approach. Another work in this area, Personalized PageRank [5, 4] is discussed in Sec. 2.3.

The purpose of this paper is to introduce a stable balanced reputation propagation scheme on arbitrary topologies. Section 2 introduces definitions and

the basic event counting scheme. Section 3 extends the model to the case of interfering fuzzy contexts (the "reputable chef" case). For basic application considerations (i.e. deriving reputation of previously unknown entities from recommendations made by known ones, using coarsened reputation/opinion maps) see [28].

2 Measuring reputation; definitions

2.1 General considerations

Mui [13] describes reputation typology including the following aspects: context, personalization, individual or group, direct or indirect (the latter includes prior-derived, group-derived and propagated). This paper discusses personalized (n^2) reputation regarding some fixed context. Individual and group, direct and indirect flavours of reputation are defined via the basic uniform notion of *responsibility* for elementary events. A reputation context is represented as a set of all relevant (past) events $\mathbb{U}$ plus compliance requirements. Generally, we must consider $\mathbb{U}(t)$ but for the sake of simplicity I will focus on a static model. Propagation of reputation is performed by a social network of recommendations derived from the same notion of responsibility.

An irresponsible recommendation and imposed responsibility (both sound like definitions of a self-assured power) are of no interest to us. There are no distinction between groups and individuals; I use the same word "entities" to describe them.

2.2 Reputation is ...

> A reputation is an expectation about an agent's behavior based on information about or observations of its past behavior. [3]

So, a reputation is based on a responsibility, i.e. an association between events (behavior elements) and entities (agents). A reputation can not exist in anonymized environments. Speaking in terms of the formal model being explained a definition of a reputation is:

Definition 1. *A reputation is an expectation that a compliance of some future event will be near to an average compliance level of past events by the same responsible entities.*

Requirements for compliance are fixed. The simplest example is "the mail (event) is a spam (non-compliant)". So, an elementary (simple) event ε initiated by entity e, $\varepsilon \in \mathrm{E}_e$, may be valued by another entity v as $\rho_v(\varepsilon) \in [0, 1]$

Our compliance expectation on a future event is based on compliance of past events by the same responsible entities, $\rho(\varepsilon) = \rho(\bigcup E_{e_i})$. A reputation of an entity is a compliance expectation on events initiated by that entity: $\rho(e) = \rho(\varepsilon)$. Considering the initiator only (i.e. one fully responsible entity) and assuming events to be of equal value (not distinctively priced) we have

$$\rho_v(e) = \rho_v(\mathrm{E}_e) = \frac{\sum_{\varepsilon \in \mathrm{E}_e} \rho_v(\varepsilon)}{|\mathrm{E_e}|} \tag{1}$$

where $|\mathrm{E_e}|$ is the number of elements (events), $\mathrm{E_e}$ is generally a set of events which affect reputation of e (it is equal to the set of events initiated by e here). We will distinct E_e as a set of known events and $\mathbb{E}_e$ as a set of all such events whether known or unknown to us. (This is the last time I mention $\mathbb{E}$ in this paper.)

2.3 Recommendation: responsibility for other's events

Definition 2. *A recommendation is an expressed opinion of an entity that some another entity is reputable which opinion the recommender is responsible for.*

Full responsibility for an event mean that the event will be included into the entity's relevant event set, thus affecting the reputation of the entity. A reputation of a recommending entity will be affected by any event that affects a reputation of recommended one.

It is useful, if recommendation could be of different certainty ("cautious", fuzzy, $0 < c < 1$), so a weight of a recommended event will be lesser than weights of events initiated by the entity itself (or, another way, the recommended event belongs to the event set of the recommender in a fuzzy way having membership degree $\mu_E = c$). To migrate to fuzzy sets a compliance-of-a-set function (Eq. 1) have to be generalized; it will be equal to a weighted mean (centroid):

$$\rho(\mathrm{E}) = \frac{\sum_{\varepsilon \in \mathrm{E}} c_\varepsilon \rho(\varepsilon)}{|\mathrm{E}|}, \text{ where } |\mathrm{E}| = \sum_{\varepsilon \in E} c_\varepsilon \tag{2}$$

Discounted inclusion $\subset_c$ is an operator further used to express recommendation and, therefore, fuzzy inclusion of a recommended event set into the recommender's set of responsibility. $\mathrm{E}_e \subset_c \mathrm{E}_r$ if entity r recommends entity e with certainty c, so $\forall \epsilon \in \mathrm{E}_e : \mu_{\mathrm{E}_r}(\epsilon) \geq c \cdot \mu_{\mathrm{E}_e}(\epsilon)$. An operation of set discounting $c\mathrm{E}$ is defined as follows: $\mu_{c\mathrm{E}}(\epsilon) = c\mu_{\mathrm{E}}(\epsilon)$. So, $\mathrm{E}_e \subset_c \mathrm{E}_r \Leftrightarrow c\mathrm{E}_e \subset \mathrm{E}_r$ where the subsethood on the right side is the original fuzzy containment by Zadeh: $A \subset B$ is true if $\mu_A(\epsilon) \leq \mu_B(\epsilon)$ for every ϵ. (Discounted inclusion correlates with Goguen implication; generally, it may be understood as a less common denominator statement on a class of entities permitting useful implications for any given entity inside the class.)

Note: this way we define a closure model using multiplication as a concatenation function and maximum as an aggregation function. This combination has a feature of *strong global invariance*[15]. It is different from Personalized PageRank which uses sum for aggregation. This difference may be described as "recommendation" (this model) vs. "voting" (PPR). Practical consequences of this feature still have to be evaluated.

So, what the entity is responsible for? A set of all events that affect a reputation of an entity e was denoted as E_e. It contains events initiated by e (membership degree 1.0) as well as events initiated by recommended entities including those recommended by recommended ones, transitively (membership degree is equal to certainty c or $c_1c_2...c_n$ for transitive cases). Recursive formulae:

$$E_e = O_e \cup \underline{R}_e = O_e \ \cup \ \bigcup c_{er_i} E_{r_i} \tag{3}$$

where
c_{er_i} is a certainty of recommendation of r_i by e;
O_e - a set of events, initiated by e;
$\underline{R}_e$ - appropriately discounted events by entities recommended by e.

So, according to Definitions 1 and 2, we expect events initiated by some known entity e to have compliance level of $\rho(E_e)$ as of Eq. 3.

What about recommended entities? Due to Definition 2 a recommender entity is responsible for events initiated by recommended ones. So, according to Definition 1, a reputation of a recommended entity depends on reputations of recommenders (i.e. events by recommended entities belong to wider sets than own event set of the initiator). Thus, for recommended entities we have:

$$E_e \ = \ O_e \cup \underline{R}_e \cup \overline{R}_e \ = \ O_e \ \cup \ \bigcup c_{er_i} E_{r_i} \ \cup \ \bigcup c_{m_j e} E_{m_j} \ \ ^{1} \tag{4}$$

where m_j are recommender entities. As a result, "an echo" of an event traverses edges in either direction, because everything that affects a reputation of a recommender, also affects reputations of recommended entities and vice-versa.

3 Contexts

The model explained in Section 2 assumes some fixed context. Taking all possible contexts as a numbered set we may provide corresponding event universes and reputations for every context. Practical applications may require more flexible approach to contexts: an event may relate to a given context at some degree, also different contexts may be semantically close. This section aims to extend the model to fuzzy compliance (relevance, reputation) contexts using the same mathematical apparatus.

First, we have to extend the universe of discourse to all events, independent of context. This extended universe will be denoted as $\mathbb{A}$. Any fuzzy set $\mathbb{U}$ in universe $\mathbb{A}$ is a context.

Taking $\mathbb{U}$ as a universe of discourse, every entity defines a fuzzy subset of compliant events $\mathbb{U}_v^+$ defined by a membership function $\rho_v(\varepsilon)$. As before, E_e is a fuzzy set containing events entity e is responsible for.

[1] As far as I see, this way of discounting of recommenders' event sets does not follow from definitions, e.g. c^2E does not contradict them also. cE is chosen for the sake of symmetry and balance, to prevent reputation "xeroxes" and "laundries".

Thus, if moving to the universe of discourse $\mathbb{A}$, Eq. 2 changes as follows:

$$\rho_v(E)|_{\mathbb{U}} = \frac{\sum \mu_E(\varepsilon)\rho_v(\varepsilon)}{\sum \mu_E(\varepsilon)} = \frac{\sum c_\varepsilon \mu_{\mathbb{U}}(\varepsilon)\rho_v(\varepsilon)}{\sum c_\varepsilon \mu_{\mathbb{U}}(\varepsilon)} \tag{5}$$

where c_ε is the degree of responsibility, as before; $\mu_{\mathbb{U}}$ is the degree of relevance to the context. So, a weight of an event becomes proportional to its relevance to a context in the case of interfering fuzzy contexts.

The rationale behind Eq. 5 and migration to the narrower universe may be explained as follows: whether we are looking for "reputable chef" (i.e. cooks well) or "chef AND (generally) reputable man" (also pays taxes, etc). For the latter case logical operaions (AND/OR) are enough, while the former needs a move to the universe of cooking.

Understanding context as a semantic domain marked by a character sequence we may model it as a fuzzy set containing relevant events and so, to easily express theoretic-set relations between different domains, i.e. inclusion, equality, disjointless and other, using the relation of discounted inclusion $\subset_c$ and similar tools. E.g. (a simple crisp example) "merinos are sheep", so a reputable sheep breeder will handle merinos well. A net of such interconnected domains can hardly be called a taxonomy or topological space. So, I suggest the term "indranet" because of some Buddhist connotations. Important feature of indranets (quasi-topological spaces formed by discounted inclusion, union and intersection) is a possibility for any subset to contain all other subsets, at some degree (e.g. due to recommendation graph closure any given entity may be responsible for any event on the planet, *at some vanishing degree*). The previous example of "reputable chef" may be understood as a seeking in the indranet formed by two orthogonal dimensions: occupation taxonomy and binary notion of reputability. General search algorithms for indranets is the author's future work.

References

1. L. Page, S. Brin, et al: The PageRank Citation Ranking: Bringing Order to the Web, 1998, http://www-db.stanford.edu/ backrub/pageranksub.ps
2. Francisco Botana: Deriving fuzzy subsethood measures from violations of the implication between elements, LNAI 1415 (1998) 234-243
3. A. Abdul-Rahman, S. Hailes: Supporting trust in virtual communities, in Proceedings 3rd Ann. Hawaii Int'l Conf. System Sciences, 2000, vol 6, p. 6007
4. P.-A. Chirita, W. Nejdl, Oana Scurtu: Knowing Where to Search: Personalized Search Strategies for Peers in P2P Networks, SIGIR'04
5. G. Jeh, J. Widom: Scaling personalized web search. In proc. of WWW'2003
6. P. Resnick, R. Zeckhauser, E. Friedman, K. Kuwabara: Reputation systems, Communications of the ACM, Volume 43, Issue 12, 2000
7. Bin Yu, M.P. Singh: A social mechanism of reputation management in electronic communities. in Proc. of CIA'2000, 154-165
8. J. Klensin, RFC 3467 "Role of the Domain Name System (DNS)", 2001
9. K. Aberer, Z. Despotovic: Managing trust in a P2P information system, in proc. of CIKM'01

10. P. Resnick, R. Zeckhauser: Trust among strangers in internet transactions: empirical analysis of eBay's reputation system, Technical report, University of Michigan, 2001
11. R. Cox, A. Muthitacharoen, R.T. Morris: Serving DNS using a peer-2-peer lookup service, in IPTPS, Mar. 2002
12. L. Mui, M. Mohtashemi, A. Halberstadt: A computational model of trust and reputation, HICSS'02
13. L. Mui, PhD thesis: Computational models of trust and reputation: agents, evolutionary games and social networks, MIT, 2002, http://medg.lcs.mit.edu/ftp/lmui/
14. Bin Yu, M.P. Singh: Detecting deception in reputation management, in Proceedings of AAMAS'03
15. M. Richardson, R. Agrawal, P. Domingos: Trust management for the Semantic Web, in Proc. of ISWC'2003
16. Jennifer Golbeck, Bijan Parsia, James Hendler: Trust networks on the Semantic Web, in Proc. of CIA'2003.
17. S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina: The EigenTrust algorithm for reputation management in P2P networks, in Proceedings of the WWW'2003
18. "Lightweight MTA Authentication Protocol (LMAP) Discussion and Comparison", John Levine, Alan DeKok. Internet Draft, 2004
19. Jennifer Golbeck, James Hendler: Reputation network analysis for email filtering, for the 1st Conf. on Email and Anti-Spam, 2004
20. Jennifer Golbeck, James Hendler: Inferring reputation on the semantic web, for EKAW'04.
21. Michal Feldman, Antonio Garcia-Martinez, John Chuang: Reputation management in peer-to-peer distributed hash tables, http://www.sims.berkeley.edu/∼mfeldman/research/
22. R. Guha, R. Kumar, P. Raghavan, A. Tomkins: Propagation of trust and distrust, in Proc. of WWW'2004
23. A. Fernandes, E. Kotsovinos, S. Ostring, B. Dragovic: Pinocchio: incentives for honest participation in distributed trust management, in Proceedings of iTrust'2004
24. Philipp Obreiter: A case for evidence-aware distributed reputation systems, in Proceedings of iTrust'2004
25. Paolo Massa, Bobby Bhattacharjee: Using trust in recommender systems: an experimental analysis, in Proceedings of iTrust'2004
26. Radu Jurca, Boi Faltings: Truthful reputation information in electronic markets without independent verification, icwww.epfl.ch/publications/documents/IC_TECH_REPORT_200408.pdf
27. Web-o-Trust effort by Russell Nelson, http://www.web-o-trust.org/
28. Viktor S. Grishchenko: Redefining Web-of-trust, may be found at http://blogs.plotinka.ru/gritzko/accounting2.pdf
29. W3C recommendation: OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/