Real-world Trust Policies

Vinicius da S. Almendra[1], Daniel Schwabe[1]

(1) Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio
Rio de Janeiro - Brazil
{almendra,dschwabe}@inf.puc-rio.br

Abstract. One of the most important problems on the semantic web area is the one of trust. The growing exchange of semantic web data raises the need of policies that allow filtering out untrustworthy information. It is necessary, however, to model adequately the concept of trustworthiness, otherwise one may end up with operational trust measures that lack a clear meaning. It is also important to have a path from one's trust requirements to concrete trust policies. Our proposal is to ease the building of this path, through a representation of trust requirements grounded on a specific notion of trust and an algorithm to map this representation to trust policies. We report our ongoing effort on this direction.

1 Introduction

One of the great challenges to the semantic web is the problem of trust. Operational measures of trustworthiness are needed to separate relevant and truthful data from those that are not [2]. However, to be correctly interpreted, these measures must be linked with real-world concepts of trust. They also must meet the trust requirements of their users. Building on the trust concept found in [4], our work aims to pave the path leading from a user's trust requirements to operational trust policies that can be applied to semantic web data, while preserving the relation between the resulting policies and the trust requirements we started with. This relation is important as it enables the user to find out why a piece of data was found trustful. We're focusing on the semantic web data exchange scenario, where an agent receives some data and must decide whether or not to trust them.

Here we report our ongoing effort in this direction, which includes a model to represent trust requirements and a small test implementation, based on the Semantic Web Publishing vocabulary [3].

1.1 Related Work

In [1] we find a very similar work: a Semantic Web Browser with filtering based on trust policies. The user selects the trust policy he wishes to use and then the software filters information using that policy. Besides that, it offers explanations of why each piece of information was trusted. The difference lies in the level of abstraction. The proposal in [1] offers facilities to express trust policies as pieces of TriQL queries in such a way that it allows an explanation of why a triple was found trustful. Our work deals with representing trust requirements and translating them to trust policies that preserve real-world trust relationships.

1.2 A Motivating Scenario

The scenario we are focusing on is based upon two works: the Semantic Web Publishing scenario [3] and the DBin project [5]. The Semantic Web Publishing scenario has agents embodying two roles: of information providers and of information consumers. An information provider publishes RDF graphs; these graphs contain information and its metadata, such as provenance, publishing date etc. An information consumer gathers these graphs and decides what to do with them, provided that these graphs can be seen as claims of the information provider, rather than definitive facts. The formal meaning of these claims, that is, what statements about the world are being made, is given by a set of *accepted graphs*, which is a subset of the graphs the information consumer receives. It is assumed that the agent will act based solely on information contained in accepted graphs.

The Semantic Web Publishing proposal also enables the user to specify a *trust policy*, that is, a set of conditions that the received information should meet to be accepted. An example of a policy would be “trust all information that comes from direct friends and is about computers”.

This scenario can be integrated with the one outlined in [5]: a P2P network where people exchange RDF graphs of interest and store all the received graphs in a local database. Filtering can be applied to hide triples that do not match the user’s criteria. One use for this is the implementation of trust policies. The set of visible triples is similar to the one of accepted graphs seen above; we will name it the set of *accepted triples*.

These scenarios are only examples of possible uses of trust and trust policies within the Semantic Web context; many other scenarios are possible, such as Semantic Social Desktops.

2 A Model of Trust

2.1 The Concept of Trust

Following the ideas presented in [4] and [7], we will use the concept of trust as “knowledge-based reliance on received information”, that is, an agent decides to trust (or not) based solely on his knowledge, and the decision to trust implies the decision to rely on the truth of the received information to perform some action. We will elaborate some key aspects of this definition below.

- **Knowledge-base trust**. The agent's knowledge includes all information the agent has, which in turn includes information received from other agents and self-gathered information. We will call the subset of this information that the agent has decided to trust “trusted information”. Received information that is not trusted will be called “known information”. The decision to trust is not irrevocable: knowledge can evolve and new evidence may render formerly trusted information untrusted, and vice versa.

- **Trust as reliance**. The idea of reliance means that the agent can use the trusted information to achieve some goal, without further analysis – although this may imply running the risk of taking an inappropriate action if the information is false. For example, if an agent trusts the information that www.mybank.com is the URL of his bank, then he will send his password to this site without further checks: he relies on this information for performing financial transactions. If the agent does not have any action depending on that information, then there is no reliance attitude and the concept of trust does not apply (for example, if he has no relationship with this bank, then the information about the URL does not matter: it is not an object of reliance).
- **Reliance on information**. This concept is about trust on information, where “to trust” means “to move known information to the set of trusted information”. There are other actions that need trust, beyond accepting information: moving money from one account to another, running an unknown software, providing sensitive information to a website, granting access to an intranet etc. Nevertheless, many of these actions rely on knowledge about the action itself, the agents involved and the circumstances: a bank normally relies on a supplied account number and password to grant access to a person's account (actually, it relies on the relation between the person, an account and a password). So, the trust concept as defined here can also be applied to these actions whenever it is possible to factor out the reliance on some kind of received information. In the case of running an unknown software, the agent will trust it or not based on the information he has about that software, e.g. who obtained it, what it does, who is the publisher etc.

There are other important aspects of the concept of trust that are relevant to the proposed model:

- Trust can be seen as a relationship between two agents mediated by a goal: one trusts somebody for something [7]; in our case, trusts somebody for receiving information from him.
- Trust is subjective, that is, each agent may have a different view about what can be trusted. So, from now on we will use the term *trusting agent* to denote from whose viewpoint trust is being evaluated.
- It is of common sense that a person normally trusts himself as a provenance of information, although he might give up on this perception if someone he believes to be wiser (with respect to this particular subject matter) contradicts him. In this work we assume that the default attitude of the trusting agent is to trust everything that comes from himself.

2.2 The Trust Act

When an agent receives information, he must decide whether or not to trust it: this is the trust act [4]. To do this, the agent can use the following elements [6]:

- The context of the received information, that is, metainformation about circumstances (provenance, date, time, location, reason, relation of the provenance with

the trusting agent etc). For example, the sender, the date and the subject of an e-mail are part of its context. However, the context may include information not present in the received information but in the agent's knowledge, like the sender's job, which one might have stored in his personal agenda. This is possible when there is some kind of URI for the sender, which "links" the agent's knowledge with the received information.
- The content of the received information.
- The reputation of the source, that is, what other agents say about it.

These elements provide *information* about the received information. The trust act consists of checking whether or not these elements satisfy some conditions. One trusts a received e-mail when he knows the sender, for example. These conditions will be called *trust requirements*. Continuing the e-mail example, we can formulate the following example trust requirements when reading e-mail:
- To download an e-mail, it must be from a known source.
- To open an e-mail, it must be written in my native language.
- To run an executable attachment, it must have been sent by a close friend and must have been verified by some kind of antivirus software.

As the trust act itself relies on information, it is reasonable to require that it should be based only on trusted information. This has two important side effects:
- A source's reputation becomes part of the context, as it will be composed by trusted information (that is, trusted opinions of other agents) related to the source. Consequently, we will restrict the elements of the trust act to encompass context and content. Notice that it is possible to use untrusted information about reputation to make a trust decision: it is what happens in many reputation systems, where the score used to evaluate an agent is made from opinions of unknown agents.
- Some of the contextual information may come together with the received information. An e-mail carries information about its sender, the date it was posted etc. If the trust acts related to e-mail demands some of this contextual information, these acts will fail (that is, the e-mail will not be trusted) until the contextual information is also the subject of a trust act and gets included in the set of trusted information. So, for a trust act about the content of received information to succeed, prior trust acts about its context should have been made, otherwise the former trust act may fail due to lack of trusted information. For instance, to reason about an e-mail using the sender's name, I must trust that who claims to be the sender *is* the sender indeed.

2.3 Formalizing Trust

From the concepts presented above it is possible to define trust as a predicate over knowledge (K), provenance (p), context (c') and content (c) of information: $T(K,p,c',c)$. This predicate is defined by the set of trust requirements of the trusting agent. From now on, we will call it the *root trust predicate*. Notice that agents with the same knowledge may react differently when faced with the same information, as they may have different trust requirements.

Using the idea of trust requirements as conditions on received information, we can represent one's trust requirements using logical predicates, similarly to [8]. The trust problem then becomes the one of building the root trust predicate. Instead of tackling the problem of building a single predicate that encodes all trust requirements of an agent (a “top-down” approach), we can try to reduce the problem to building the root trust predicate as a composition of simpler predicates that can be evaluated independently. This way, the root trust predicate becomes a disjunction of these simpler predicates: when faced with some information, it will be trusted if at least one of these simpler predicates is true. We call these simply *trust predicates*.

As an example, the root trust predicate shown below states that the user will trust information if it is “good email” or “good software”.

$$T(K,p,c',c) \leftarrow GoodEmail(K,p,c',c) \vee GoodSoftware(K,p,c') \qquad \mathbf{(1)}$$

Each of the trust predicates states necessary and sufficient conditions to trust some piece of information. This can be put in evidence by breaking them into conjunctions of other trust predicates:

$$\begin{aligned} GoodEmail(K,p,c',c) \leftarrow\ & IsFromFriend(K,p) \\ & \wedge IsEmail(c) \wedge \neg Old(c') \end{aligned} \qquad \mathbf{(2)}$$

What distinguishes trust predicates from other logical predicates is that the former express meaningful conditions for trust *from the agent's viewpoint*. In the example, it does not matter how the name of a source is represented; what matters is whether or not the source is a friend.

Trust predicates should express trust conditions that make sense for the trusting agent, linking his trust-related concepts (e.g. “being a friend”, “belonging to a research group”, “being a relevant author”) to logical conditions on the information available (e.g. “being referenced in my personal FOAF file”, “having a link from a specific web page to the source's FOAF”, “appearing in the citations of more than five publications”). Another possibility is to define them in terms other trust predicates (e.g. the GoodEmail predicate in the example above: it is a trust predicate composed by other trust predicates), in this case, we will call it a *composite trust predicate*. In contrast, an *elementary trust predicate* is one that has no meaningful decomposition from the agent’s point of view and must be built by directly stating conditions on information.

To strengthen the connection with real-world trust, we will apply the formal model with the trust concept presented in [4], where the decision to trust is made based on the appreciation of three things: the subject matter (the content), the entity involved (in our case, the provenance), and the circumstances (the context). Therefore, if trust predicates can be specified to verify these three aspects, then it is possible to solve completely an arbitrary instance of the trust problem (within the limits of expressiveness of the formalisms used) by making a trust predicate that is a conjunction of those trust predicates. We will call this conjunction, namely, the triple <subject matter, entity, circumstances>, a *trust-point* [4]. Accordingly, the root trust predicate can be described as a disjunction of trust-points. In the preceding example, *GoodEmail* is a trust-point: it evaluates the entity (first predicate, *IsFriend*), the subject matter (second predicate, *IsEmail*) and the circumstances (third predicate, not *Old*). If the three trust predicates are true for the received information, then it is considered trustful. Summarizing, a trust-point is a set of conditions on *who* sent the information, on *what* is the information, and on what *circumstances* surround it. According to this model, the trust act consists of applying the root trust predicate to a known fact to determine whether or not it should be trusted.

It is important to notice that the provenance of information is part of the context in [6], whereas it is factored out in [4]. We adopted the latter approach, grounded on the idea that trust is a relationship between agents and hence the model should capture this explicitly.

2.4 An Example

Looking at the scenario presented and restricting it to the exchange of scientific information among researchers, we can identify many trust predicates, some more general (i.e., they apply to other domains), others more specific. Each predicate is followed by its name in the logical formulas:

- Trust predicates related to entities involved: “works with me” (Colleague), “is a relevant researcher” (IsRes), “is cited by other authors” (IsCited).
- Trust predicates related to the matter: “is a publication” (IsPubl), “is a website” (IsSite), “is a relevant website” (GoodSite), “is cited by a relevant website” (CiteByGoodSite), “is the contact information of a researcher” (InfoRes), “is a relevant publication” (GoodPubl), “is the contact information of a person” (IsCInfo).
- Trust predicates related to circumstances: “is recent enough” (IsRecent), “is old enough”, “is newer than my preferred publications”, “is hosted in a university” (HostUniv).

With these trust predicates, we can model the trust requirements of two hypothetical agents, John and Mary, concerning acceptance of scientific information.

$$\begin{aligned} T_{John}(K,p,c',c) &\leftarrow GoodPubl_{John}(K,p,c) \vee GoodSite(K,p,c) \\ T_{Mary}(K,p,c',c) &\leftarrow GoodPubl_{Mary}(K,p,c',c) \vee InfoRes(K,p,c) \end{aligned} \qquad (3)$$

T_{John} and T_{Mary} are the root trust predicates representing trust requirements of the agents. John is prepared to trust relevant publications and websites; Mary also is prepared to trust relevant publications (although she may have a different concept of what a good publication is, as will become clearer the example), as well as contact information of researchers. *GoodPubl*, *GoodSite* and *InfoRes* are the trust-points involved. Now we proceed to describe these trust-points using the <subject matter, entity, circumstances> template. In Table 1, each trust-point will be described in natural language, and then cast as trust predicates.

Table 1. Description of the trust-points

Trust-point	Subject Matter	Entity	Circumstances
GoodPubl$_{John}$	It is a publication and it is cited by a relevant website	It has been sent by a colleague	-
GoodPubl$_{Mary}$	It is a publication and it is cited by relevant publications	It has been sent by a researcher	It is not recent
GoodSite	It is a website	It has been sent by a researcher	It is hosted in an university
InfoRes	It is contact information of a known researcher	It has been sent by a colleague	-

Now it is possible to specify each trust-point using the trust predicates shown before:

$$\begin{aligned}
GoodPubl_{Mary}(K,p,c',c) &\leftarrow IsPubl(c) \wedge IsRes(K,p) \wedge \\
&\quad \exists s(IsCited(K,c,s) \wedge GoodPubl(K,s)) \\
&\quad \wedge \neg IsRecent(c') \\
GoodPubl_{John}(K,p,c) &\leftarrow IsPubl(c) \wedge Colleague(K,p) \wedge \\
&\quad CitedByGoodSite(K,c) \\
GoodSite(K,p,c) &\leftarrow IsSite(K,c) \wedge IsRes(K,p) \\
&\quad \wedge HostUniv(K,c) \\
InfoRes(K,p,c) &\leftarrow Colleague(K,p) \wedge \\
&\quad \exists r(IsCInfo(c,r) \wedge IsRes(K,r))
\end{aligned} \qquad (4)$$

One thing to notice is the "reuse" of trust predicates (*IsPubl*, *Colleague*, *IsRes*): we believe this decomposition allows great simplification of the process of building trust-points, as many trust decisions rely on the presence (or absence) of the same properties (for example, "being a colleague", "being a relevant author", "being cited").

2.5 A Model for the Trust Process

Having a model for the trust requirements, now we can proceed to model the dynamics of trust: how the trust acts are combined to form what we call the *trust process*, which is the process by which an agent keeps its trusted facts base with all the facts that the agent trusts, according to its trust-points, and with no facts that he does not trust.

We propose the following procedure for realizing the trust process:

1. Include in the known facts base facts contained in the received information.
2. Remove one fact from either the trusted or the known facts base that has not been analyzed yet.
3. Apply the trust act to it.
4. If the fact tested is found trustful, then include it in the trusted facts base. If not, include it in the known facts base.
5. Go back to step 1 until all facts (either known or trusted) have been analyzed.
6. If no fact changed its status (from known to trusted or vice versa), the process ends; otherwise, restart the process from step 2.

This procedure is depicted in figure 1. Notice that the trust act uses only the set of trusted facts and the trust-points to decide the trustfulness of a fact.

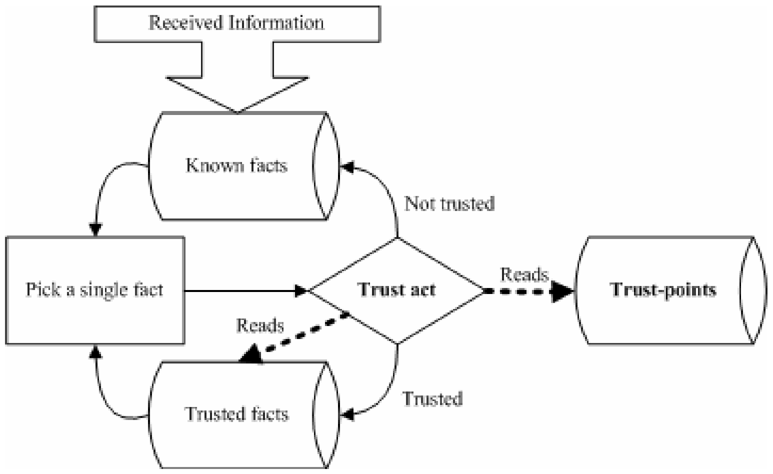

Fig. 1. Depiction of the trust process

An iterative procedure is used, as the conditions that each trust-point states to accept a set of facts (for a given provenance) are fulfilled depending on the presence (or absence) of other trusted facts. So, to trust a fact, it may be necessary that other facts have already been trusted, which also means that trusting a fact may lead other facts to be trusted. The same happens when a fact loses its trust status: other facts that depended on this one may also become untrusted.

The last step of the procedure is justified by the functional nature of the trust act: if there are no changes in the inputs, then nothing will change if it is applied again.

The dependence among facts shown above presents an interesting property of the model: the interplay between trust-points. The dependence that exists in practice among facts reveals the intrinsic dependence among trust-points. When a trust-point is added, it may implicitly enter in a chain of trust-points. These implicit chains of trust-points resemble the trust policies that are explicitly defined in other approaches like [8]. We say that the model allows the implicit definition of arbitrarily complex trust policies through the use of its building blocks, the trust-points.

2.6 Trust Transitivity

One often-used property of the trust relationship is the transitivity [9, 10]. It is commonly used jointly with trust degrees to represent trust relationships as weighted graphs, where the weight is the degree of trust. Trust propagation algorithms are used to infer the degree of trust between unrelated nodes (that is, nodes with no direct edge between them).

Our approach does not yet use transitivity of trust in the model: it does not support trust-point inference through the use of trust relationships. In other words, trust-points are independent: the fact that one agent trusts another on some aspect does not influence the agent's trust on a third agent.

However, the model does provide transitivity *de facto* in a scenario of information sharing where agents use the proposed trust model and each one of them only shares information that they trust. To see how this happen, we must derive a trust graph, where each trust-point originates an edge from the trusting agent to the trusted agent (the provenance); this edge's "weight" is the subject matter of the trust-point. If we merge the trust graphs of several agents, one might find a path in which all edges have the same subject matter, e.g. John trusts Mary on finding papers, Mary trusts Daniel on finding papers and Daniel trusts Mark on finding papers. In the example, when Mark finds an interesting paper, it will eventually end up being trusted by all the agents in the path (see Fig. 2).

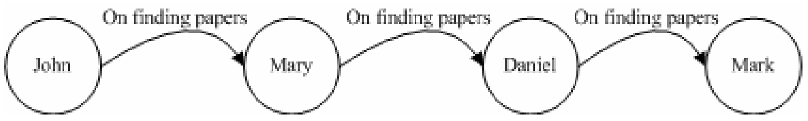

Fig. 2. *De facto* trust transitivity

2.7 Degrees of Trust

It is a common perception that trust has degrees: one can trust someone more than someone else [7]. In formal terms, there is an order on the relation of trusting agents and information sources. Some models try to capture this order through the assignment of trust ratings to trust relationships [10]. This assignment is made for some pairs of agent-source and then the remaining ratings are inferred [9].

Although the proposed model does not use trust degrees, it does capture the ordering of trust relationships based on the idea that to trust more one agent means to trust him over a larger scope of "things". A single trust-point's subject matter gives a set of trusted facts for some provenance. Given a trusting agent T_a, another agent A, we can group all trust-points of T_a that include A in their provenance. From this set of trust-points we can extract the set of subject matters on which T_a trusts A, which gives the *extent of trust of T_a with respect to A*. To visualize this, we can look again to the trust graph originated by the trust-points: each pair of nodes can have one or more edges, one for each subject matter. The set of edges between two nodes gives the extent of the trust relationship, as exemplified in Fig. 3, which shows the extents of trust of John with respect to Mary (accepting web sites, Semantic Web papers and rock music) and Bob (accepting web sites and antivirus software). Formally, we can say that the trust predicate associated with a subject matter *entails* the set of facts that will be trusted. Then, we can say that the extent of trust of T_a with respect to A entails the set of all facts that T_a trusts when coming from A.

Now the problem is reduced to finding an order on the set of extents. A partial ordering on it is given by the entailment relation: an extent A is strictly greater than the extent B if A entails B and B does not entail A. Loosely speaking, an extent is greater than other if it *contains* the other extent. If there is no entailment relation between two extents, then it is not possible to order them.

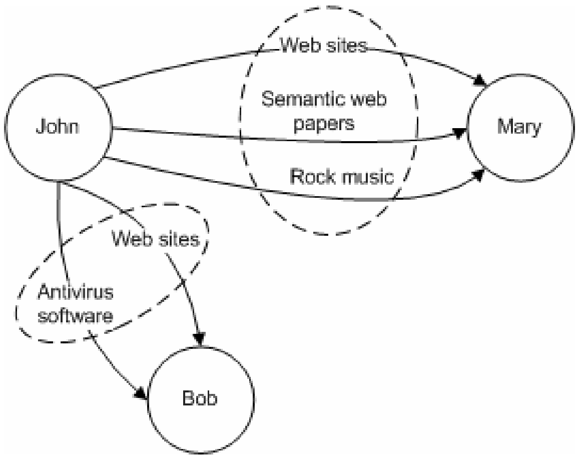

Fig. 3. John is the trusting agent. The dashed lines show the extents of trust of John with respect to Mary and Bob.

2.8 Applying the Model to the Motivating Scenario

The above model of trust can be applied to the scenario of interest with the following mappings:

- A fact is an RDF triple pertaining to a named graph [3]. We used named graphs in order to support attachment of provenance and contextual information to graphs (see below).
- The trusted facts are the accepted triples.
- The trusting agent is the information consumer.
- The circumstances are facts whose subject is the context of a triple, that is, its surrounding named graph. The provenance is a particular circumstance that will be treated separately, as justified before.

- The Semantic Web Publishing vocabulary [3] provides means to attach provenance information to graphs, establishing a relation between an agent and a graph. This relation can be of assertion, quoting, denial etc. So, a triple's provenance is the entity asserting this triple's graph.
- The subject matter *entails* a set of triples that the trusting agent relies on (for a particular provenance). The subject matter "receiving academic articles" will entail all triples that, in the domain theory of the agent, assert that something is an academic article. Notice that this relation can (and normally will) be described intensionally. In the preceding example, the entailment relation can be stated using an RDF property whose domain is the set of academic articles.
- The trust policy of an agent for receiving information is given by its set of trust-points. The trust process presented enforces this policy, separating reliable facts from facts that are just known.

3 Prototype Implementation

We tested the preceding ideas in a prototype solution aimed to partially implement the trust process presented above. We used named graphs to store triples and TriQL (the extension of RDQL for named graphs) to represent (and apply) trust predicates. Currently we do not implement neither composite nor negated trust predicates. We also do not support transparently blank nodes due to limitations of the underlying implementations used. Currently all blank nodes are transformed into fake URI nodes during the prototype execution.

The trust-points were expressed using a simple ontology, mirroring the model's structure: each trust-point is composed by one or more trust predicates, which may be predicates on the subject matter or on the provenance. We do not provide separate predicates for circumstances yet, although they can be implemented as subject matter predicates without loss of expressiveness. The elementary trust predicates are expressed as graph patterns and sentence patterns or URI, depending on the type of predicate (subject matter or provenance). Follows a sample of the trust-points and trust predicates we used in our tests. The trust-points are represented using N3 format.

```
# Describes the trust-point GoodSite

ex:goodSite a trust:TrustPoint;

      # Provenance must be a researcher
      trust:provenance trustpred:isResearcher;

      # Subject matter must be a web site
      trust:subjectMatter trustpred:isSite;

      # The site must be hosted on a university
      trust:subjectMatter trustpred:isHostedOnUniv .
```

```
# Describes the trust-point GoodPubl_John

ex:GoodPublJohn a trust:TrustPoint;

      # Provenance should be a John’s colleague
      trust:provenance trustpred:isColleagueJohn;

      # Subject matter must be a publication and
      # must be cited by a good web site
      trust:subjectMatter trustpred:isPublication;
      trust:subjectMatter trustpred:citedByGoodSite .


# Describes the trust predicate IsRes

trustpred:isResearcher a trust:ElementaryPredicate;
      trust:graphPattern
            "?g (?GRAPH swp:assertedBy ?g .
                 ?g swp:authority ?ENTITY)
             (?ENTITY foaf:member ?u)
             (?u rdf:type ex:University)" .

# Describes the trust predicate isPubl

trustpred:isPublication a trust:ElementaryPredicate;

      # This predicate is described as a sentence
      # pattern. It will allow sentences whose
      # predicate is dc:type. This is the trusting
      # agent’s view of what is a publication.
      trust:sentencePattern
            "?ANYTHING dc:type ?ANYTHING" .
```

The graph patterns are used to express conditions on the set of trusted facts, that is, what must be already known to trust a triple. They follow TriQL syntax and may use variables. Each trust-point is converted to a TriQL query, which is applied to the set of trusted facts. There are four special variables, whose values are bound before running the queries: GRAPH, SUBJ, PRED and OBJ, which are bound respectively to the graph name, the subject of the triple being tested, its predicate and its object. These variables allow the trust predicates to test relationships between the triple being analyzed with the trusted facts. The graph patterns of all trust predicates of a trust-point are concatenated to build the query.

The sentence patterns restrict the valid values on each of the triple’s components. If the agent does not wish to restrict some of them, he may use the special variable ANYTHING. These restrictions appear in the query as conditions on the variable’s values.

To deal with provenance information, the prototype recognizes the special variable ENTITY and allows the trust predicate to specify a concrete URI value for it. The trust predicate is responsible for providing a graph pattern that binds this variable to the node that represents the provenance of the triple being tested.

The trust-point *GoodSite* shown above is translated to the following TriQL query that, given a triple, rules it out if it does not match the conditions imposed by the trust-point.

```
SELECT * WHERE ?GRAPH (?SUBJ ?PRED ?OBJ)
                      (?SUBJ ex:hostedOn ?u)
                      (?u rdf:type ex:University)
                  ?g  (?GRAPH swp:assertedBy ?g .
                       ?g swp:authority ?ENTITY)
                      (?ENTITY foaf:member ?u)
                      (?u rdf:type ex:University)
           AND ?PRED eq rdf:type
           AND ?OBJ eq ex:Website
```

Once the trust-points are translated into queries, the prototype cycles through the set of triples not yet trusted, following the trust process defined earlier. Currently it operates on a static knowledge-base, that is, no new facts can be added during its execution. This restriction will be removed in future developments.

We tested the prototype with some hand-made sample data and it performed as expected, being capable of implement and enforce the trust policies expressed by the sample trust points presented. Follows the output of the prototype in one of the tests, showing which triples got trusted in which cycle. The name of the trust-point who allowed the inclusion of the fact is between angle brackets before each sentence.

```
Loading knowledge base...
Loading trust-points...
Running trust-point engine...

====> Cycle 1

+ {goodProvenance1} ex:JohnWarrant  swp:assertedBy
ex:JohnWarrant
+ {goodProvenance1} ex:AnnWarrant  swp:assertedBy
ex:AnnWarrant
+ {goodProvenance3} ex:JohnWarrant  swp:authority
ex:John
+ {goodProvenance3} ex:AnnWarrant  swp:authority
ex:Ann
+ {selfTrust} ex:JohnData  swp:assertedBy
ex:JohnWarrant
+ {selfTrust} ex:John  foaf:knows  ex:Ann
+ {selfTrust} ex:John  foaf:name  "John M."
+ {selfTrust} ex:puc  rdf:type  ex:University
+ {selfTrust} ex:Ann  foaf:member  ex:puc
+ {selfTrust} ex:swsite  ex:cites  ex:book
+ {hosting} ex:swsite  ex:hostedOn  ex:puc
+ {goodProvenance2} ex:AnnData  swp:assertedBy
ex:AnnWarrant

====> Cycle 2

+ {goodSite} ex:swsite  rdf:type  ex:Website

====> Cycle 3

+ {GoodPublJohn} ex:book  dc:type  ex:Book

====> Cycle 4

FINISHED
```

4 Conclusions

Our goal was to build a formalism to capture, represent and apply trust requirements of an agent in the scenario of Semantic Web data exchange, while preserving the real-world semantics of trust. This was done using the trust-point concept as a unit comprising all information needed to decide the trustfulness of received information. The composition of trust-points yields trust policies that can be realized using the trust process proposed. We presented a partial prototype implementation fulfilling this trust process, using TriQL queries to implement the trust policies.

Differently from Bizer's work [1], where each policy must specify all the conditions the triples must fulfill to be accepted, in the proposed model the trust policies can be built incrementally, as each trust-point can be specified independently from the others, while cooperating with them in each trust act. The addition of new trust-points enriches the resulting trust policies. In fact, these trust policies emerge from the interactions between the different trust-points. We believe this represents more realistically how trust acts occur in the real-world.

The next steps in this work include a deeper study of the proposed formalism in order to evaluate its properties (expressiveness and computational complexity, among others) and to see how it compares to other existing models. We also wish to complete the prototype's implementation and use it in more realistic scenarios, including Social Semantic Desktops and P2P networks.

5 References

1. Bizer, C., Cyganiak, R., Maresch, O., Gauss, T.: TriQL.P - Trust Policies Enabled Semantic Web Browser. http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQLP/browser/
2. Guha, R.: Open Rating Systems. 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web (2004).
3. Carroll, J. J., Bizer, C., Hayes, P., Stickler, P.: Named Graphs, Provenance and Trust. Technical report HPL-2004-57 (2004).

4. Gerck, E.: Toward Real-World Models of Trust: Reliance on Received Information. http://www.safevote.com/papers/trustdef.htm.
5. Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F.: The DBin Semantic Web platform: an overview. http://www.instsec.org/2005ws/papers/tummarello.pdf.
6. Bizer, C., Oldakowski, R.: Using Context- and Content-Based Trust Policies on the Semantic Web. In: International World Wide Web Conference (2004).
7. Castelfranchi, C., Falcone, R.: Social Trust: A Cognitive Approach. In: Castelfranchi, C.; Yao-Hua Tan (Eds.): Trust and Deception in Virtual Societies. Springer-Verlag (2001).
8. Nejdl, W., Olmedilla, D., Winslett, M.: PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. Workshop on Secure Data Management in a Connected World (SDM'04) in conjunction with 30th International Conference on Very Large Data Bases, Aug.-Sep. 2004, Toronto, Canada
9. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of Trust and Distrust. In: International World Wide Web Conference (2004).
10. Golbeck, J., Parsia, B., Hendler, J.: Trust Networks on the Semantic Web. Proceedings of Cooperative Intelligent Agents (2003), Helsinki, Finland.