The REWERSE View on Policies

P.A. Bonatti, G. Antoniou, M. Baldoni, C. Baroglio, C. Duma, N. Fuchs, A. Martelli, W. Nejdl, D. Olmedilla, J. Peer, V. Patti, and N. Shamheri

REWERSE (REasoning on the WEb with Rules and SEmantics)
WG-I2: *Policy specification, composition, and conformance*

Abstract. In this position paper we outline the vision adopted by the working group on policies of the EU FP6 Network of Excellence REWERSE, IST-2004-506779.

Keywords: Integrated heterogeneous policies, Cooperative policy enforcement, Lightweight trust, Trust management, Natural language interfaces, Explanation mechanisms.

1 Introduction

REWERSE (REasoning on the WEb with Rules and SEmantics) is one of the two european networks of excellence on the semantic web funded by the European Union within the 6th framework program.[1] The focus of REWERSE is on lightweight knowledge representation and reasoning, based as much as possible on rule-based languages because of their low computational complexity.

One of REWERSE's working groups, WG I2, is expressly devoted to *policy specification, composition, and conformance*. The members of WG I2 have identified in policies one of the most interesting areas for applying semantic web ideas.

Policies are pervasive in web applications. They play crucial roles in enhancing security, privacy, but also service usability. They may determine the success of a web service (or its failure). A user will not be able to benefit of the protection mechanisms of its system until she understands and is able to personalize the policies applied by the system. Simlarly, the facilities of a web service will not be fully available to its customers unless they understand the policies applied by the system (access control policies, privacy policies, and business rules, at least).

The vision of WG I2 can be summarized by the following list of strategic goals and lines of research:

– We adopt a *broad notion of policy*, encompassing not only access control policies, but also privacy policies, business rules, quality of service, etc. We believe that all these different kinds of policies should eventually be integrated into a single framework.

[1] REWERSE has 27 academic and industrial participants distributed across 14 european countries. The network officially started on March 1, 2004. More information on `http://www.rewerse.net`.

- *Strong and lightweight evidence*: Policies make decisions based on properties of the peers interacting with the system. These properties may be strongly certified by—say—cryptographic techniques, or be reliable to an intermediate degree, where evidence gathering and validation are easier (lightweight evidence). A flexible policy framework shall merge the two forms of evidence to meet the efficiency and usability requirements of web applications.
- The above two points imply that trust negotiation, reputation models, business rules, and action specification languages should be integrated into a single framework, to some extent. It is crucial to find the right tradeoff between generality and efficiency. *So far, no framework has tried to merge all these aspects together into a coherent system.* This is one of the hard challenges of WG I2.
- *Automated trust negotiation* (ATN)—adapted to other forms of negotiation—is one of the main ingredients that we use to make heterogeneous peers effectively interoperate. Therefore we are actively contributing to the advances in the area of *trust management*.
- By *lightweight knowledge representation and reasoning* we do not only refer to computational complexity; we mean also reducing the effort to specialize our general frameworks to specific application domains; and we mean that our tools should be easy to learn and use for common users, with no particular training in computers or logic. We regard these properties as crucial for the success of a semantic web framework.
- The last issue cannot be tackled simply by adopting a rule language. We are working at a *controlled natural language syntax for policy rules*, to be translated by a parser into the internal logical format.
- *Cooperative policy enforcement*: A secure cooperative system should (almost) never say *no*. Web applications need to help new users in obtaining the services that the application provides—potential customers should not be discouraged. When the prerequisites for accessing a service are not met, the web application should better explain what is missing and help the user in obtaining the required permissions.
- As part of cooperative enforcement, advanced *explanation mechanisms* should be developed to help users in understanding policy decisions and obtaining the permission to access the desired service.

In the rest of this paper we expand on the above issues and point out what we regard as interesting research directions.

2 A broad notion of policy

Policies are pervasive in all web-related contexts. Access control policies are needed to protect any system open to the internet. Privacy policies are needed to assist users while they are browsing the web and interacting with web services. Business rules (that in the view of WG I2 are just another kind of policy) specify which conditions apply to each customer of a web service. Other policies specify

constraints related to Quality of Service (QoS). In E-government applications, visas and other documents are released according to specific eligibility policies. Of course this list is not exhaustive, and is limited only by the class of applications that can be deployed in the world wide web.

Note that most of these policies make their decisions based on similar pieces of information—essentially, properties of the peers involved in the transaction. For example, age, nationality, customer profile, identity, and reputation may all be considered both in access control decisions, and in determining which discounts are applicable (as well as other eligibility criteria). Then it is appealing to integrate these kinds of policies into a coherent framework, so that (i) a common infrastructure can be used to support interoperability and decision making, and (ii) the policies themselves can be harmonized and synchronized.

In the general perspective depicted above, policies may also establish that some events must be logged (audit policies), that user profiles must be updated, and that when a transaction fails, the user should be told how to obtain missing permissions. In other words, policies may specify *actions* whose execution may be interleaved with the decision process. Such policies are called *provisional policies*.

Then, in our view, *policies act both as decision support systems and as declarative behavior specifications*. An effectively user-friendly approach to policy specification would give common users (with no training in computer science or logic) a better control on the behavior of their own system (see the discussion in Section 5).

Of course, the extent to which this goal can be actually achieved depends on the policy's ability of *interoperating* with legacy software and data—or more generally, with the rest of the system. Then a policy specification language should support suitable primitives for interacting with external packages and data in a flexible way.

The main challenges raised by the above discussion are the following:

- Harmonizing security and privacy policies with business rules, provisional policies, and other kinds of policy is difficult because their standard formalizations are based on different derivation strategies, and even different reasoning mechanisms, sometimes (cf. Section 4.3). Deduction, abduction, and event-condition-action rule semantics need to be integrated into a coherent framework, trying to minimize subtleties and technical intricacies (otherwise the framework would not be widely accessible to common users).
- The interactions between a rule-based theory and "external" software and data has been extensively investigated in the framework of logic-based mediation and logic-based agent programming [11, 10]. However, there are novel issues related to implementing high-level policy rules with low-level mechanisms such as firewalls, web server and DBMS security mechanisms, operating system features etc., that are typically faster and more difficult to bypass than rule interpreters [8]. A convincing realization of this approach might boost the application of the rich and flexible languages developed by the security community.

3 Strong and lightweight evidence

There exist currently two different major approaches for managing trust: policy-based and reputation-based trust management. The two approaches have been developed within the context of different environments and targeting different requirements. On the one hand, policy-based trust relies on objective "strong security" mechanisms such as signed certificates and trusted certification authorities (CA hereafter) in order to regulate the access of users to services. Moreover, the access decision is usually based on mechanisms with well defined semantics (e.g., logic programming) providing strong verification and analysis support. The result of such a policy-based trust management approach usually consists of a binary decision according to which the requester is trusted or not, and thus the service (or resource) is allowed or denied. On the other hand, reputation-based trust relies on a "soft computational" approach to the problem of trust. In this case, trust is typically computed from local experiences together with the feedback given by other entities in the network. For instance, in eBay buyers and sellers rate each other after each transaction. The ratings pertaining to a certain seller (or buyer) are aggregated by the eBay's reputation system into a number reflecting seller (or buyer) trustworthiness as seen by the eBay community. The reputation-based approach has been favored for environments, such as Peer-to-Peer or Semantic Web, where the existence of certifying authorities could not be always assumed but where a large pool of individual user ratings is often available.

Yet another approach—very common in today's applications—is based on forcing users to commit to contracts or copyrights by having users click an "accept" button on a pop-up window. This is perhaps the lightest approach to trust, that can be generalized by having users utter *declarations* (on their e-mail address, on their preferences, etc.) e.g. by filling an HTML form.

Real life scenarios often require to make decisions based on a combination of the above approaches. Transaction policies must handle expenses of all magnitudes, from micropayments (e.g. a few cents for a song downloaded to your iPod) to credit card payments of a thousand euros (e.g. for a plane ticket) or even more. The cost of the traded goods or services typically contributes to determining the risk associated to the transaction and hence the trust needed for performing it.

Strong evidence is generally harder to gather and verify than lightweight evidence. Sometimes, a "soft" reputation measure or a declaration (in the sense outlined above) is all one can obtain in a given scenario. We strongly believe that the success of a trust management framework can be determined by the ability of *balancing trust levels and risk levels* for each particular task supported by the application. So we add the following items to the list of interesting research directions:

- How should the different forms of trust be integrated? A first proposal can be found in these proceedings (see the paper by Bonatti, Duma, Nejdl, Olmedilla, and Shahmehri). However, new reputation models keep on being introduced, and there is a large number of open research issues in the

reputation area (e.g., vulnerability to coalitions). Today, it is not clear which of the current approaches will be successful and how the open problems will be solved (this is why our current proposal aims at maximal modularity in the integration of numerical and logical trust).
– How many different forms of evidence can be conceived? In principle, properties of (and statements about) an individual can be extracted from any—possibly unstructured—web resource. Supporting such a variety of information in policy decisions is a typical semantic web issue—and an intriguing one. However, such general policies are not even vaguely as close to become real as the policies based on more "traditional" forms of evidence (see the discussion in the next section).

4 Trust management

4.1 Some history

During the past few years, some of the most innovative ideas on security policies arose in the area of *automated trust negotiation* [1, 2, 5, 6, 12–15]. That branch of research envisaged peers that automatically negotiate credentials according to their own declarative, rule-based policies. Rules specify for each resource or credential request which properties should be satisfied by the subjects and objects involved. Then, at each negotiation step, the next credential request is formulated essentially by *reasoning* with the policy, e.g. by inferring implications or computing abductions.

Since year 2000 there exist frameworks where credential requests are formulated by exchanging *sets of rules* [2, 5]. Requests were formulated *intensionally* in order to express compactly and simultaneously all the possible ways in which a resource can be accessed—thereby shortening negotiations and improving privacy protection (because peers can choose the best option from the point of view of sensitivity). Intuitively, it is not appealing to request "*an ID and a credit card*" by enumerating all possible pairs of ID credentials and credit card credentials; it seems much better to *define* what IDs and credit cards are and send the definition itself. Another peer may use it to check whether some subset of its own credentials fulfills the request. This boils down to gathering the relevant concept definitions in the policy (so-called *abbreviation rules*) and sending them to the other peer that reasons with those rules locally.

In other words, in [2, 5] *peers communicate by sharing their ontologies*. Interestingly, typical policies require peers to have a common a priori understanding only of the predicate representing credentials and arithmetic predicates, because any other predicate can be understood simply by sharing its definition. Therefore, the only nontrivial knowledge to be shared is the X.509 standard credential format. In this framework, interoperability based on ontology sharing is already at reach! This is one of the aspects that make policies and automated trust negotiation a most attractive application for semantic web ideas.

Another interesting proposal of [5] is the notion of *declaration*, that has already been discussed in Section 3. This was the first step towards a more flexible

and lightweight approach to policy enforcement, aiming at a better tradeoff between protection efforts and risks.

This framework was chosen as the starting point for the work of WG I2, because according to [9] it was still one of the most complete trust negotiation systems in 2002. The major limitation was the lack of distributed negotiations and credential discovery, which are now supported as specified in [3, 2]. As we already pointed out, a first approach at integrating crisp and soft notions of trust is described in [3] and in these proceedings.

4.2 Negotiations

In response to a resource request, a web server may ask for some credentials, proving that the client can access the resource. However, the credentials themselves are sensitive resources, in general. So the two peers are in a completely symmetrical situation: the client, in turn, may ask the server for credentials (say, proving that it participates into the Better Business Bureau program) before sending off the required credentials. Each peer decides how to react to incoming requests according to a local policy, which is typically a set of rules written in some logic programming dialect. As we already pointed out, requests are formulated by selecting some rules from the policies.

This basic schema has been refined along the years taking several factors into account [1, 2, 5, 6, 12–15].

First, policy rules may possibly inspect a *local state* (such as a legacy database) that typically is not accessible by the other peers. In that case, in order to make rules intelligible to the recepient, they are first partially evaluated w.r.t the current state.

Second, *policies themselves are sensitive resources*, therefore not all relevant rules are shown immediately to the peer. They are first filtered according to policy release rules; the same schema may be applied to policy release rules themselves for an arbitrary but finite numer of levels.

As a consequence, some negotiations that might succeed, in fact fail just because the peers do not tell each other what they want. The study of methodologies and properties that guarantee negotiation success (when appropriate) is an interesting open research issue.

Moreover, *credentials are not necessarily on the peer's host*. It may be necessary to locate them on the network [7]. As part of the automated support to *cooperative enforcement*, peers may give each other hints on where a credential can be found [16].

There are further complications related to actions (cf. Section 4.3). In order to tune the negotiation strategy to handle all these aspects optimally, PROTUNE—the core policy language of REWERSE—supports a *metapolicy language* [3, 2] that specifies which predicates are sensitive, which are associated to actions, which peer is responsible for each action, where credentials can be searched for, etc., thereby guiding negotiation in a declarative fashion and making it more cooperative and interoperable. Moreover, the metapolicy language can be used

to instantiate the framework in different application domains and link predicates to the ontologies where they are defined.

4.3 Provisional policies

Policies may state that certain requests or decisions have to be logged, that the system itself should search for certain credentials, etc. In other words, policy languages should be able to specify *actions*. Event-condition-action (ECA) rules constitute one possible approach. Another approach, supported by the current core policy language of REWERSE, consists in labelling some predicates as *provisional*, and associating them to actions that (if successful) make the predicate true [3, 2]. It may also be specified that an action should be executed by some other peer; this results in a request.

A cooperative peer tries to execute the actions under its responsibility whenever this helps in making negotiations succeed. For example, provisional predicates may be used to encode business rules. The next rule (formulated in PROTUNE's language) enables discounts on low_selling articles in a specific session:

$$\begin{aligned}&\texttt{allow}(Srv) \leftarrow \ldots, \texttt{session}(ID),\\ &\quad \texttt{in}(X, \texttt{sql:query}(\texttt{'select} * \texttt{from low\_selling'}),\\ &\quad \texttt{enabled}(\texttt{discount}(X), ID)\,.\end{aligned}$$

Intuitively, if $\texttt{enabled}(\texttt{discount}(X), ID)$ is not yet true but the other conditions are verified, then the negotiator may execute the action associated to `enabled` and the rule becomes applicable (if $\texttt{enabled}(\texttt{discount}(X), ID)$ is already true, no action is executed). The (application dependent) action can be defined and associated to `enabled` through the metapolicy language of PROTUNE. With the metalanguage one can also specify when an action is to be executed.

Some actions would be more naturally expressed as ECA rules. However, it is not obvious how the natural bottom-up evaluation schema of ECA rules should be integrated with the top-down evaluation adopted by the current core language. The latter fits more naturally the abductive nature of negotiation steps. The integration of ECA rules in the core policy language is one of the open issues in REWERSE's agenda.

4.4 Stateful vs. stateless negotiations

The negotiations described above are in general stateful, because (i) they may refer to a local state—including legacy software and data—and (ii) the sequence of requests and counter requests may become more efficient if credentials and declarations are not submitted again and again, but are rather kept in a local negotiation state.

However, negotiations are not *necessarily* stateful:

- the server may refuse to answer counter-requests, or—alternatively—the credentials and declarations disclosed during the transaction may be included in every message and need not be cached locally;

– the policy does not necessarily refer to external packages.

In other words, stateless protocols are just special cases of the frameworks introduced so far. Whether a stateless protocol is really more efficient depends on the application. Moreover, efficiency at all costs might imply less cooperative systems.

The question is: *are stateful protocols related to scalability issues?* We do not think so. The web started as a stateless protocol, but soon a number of techniques have been implemented to simulate stateful protocols and transactions in a number of real world applications and systems, capable of answering a huge number of requests per time unit. We must observe that if the support for stateful negotiations had been cast into http, then probably many of the intrinsic vulnerabilities of simulated solutions (like cookies) might have been avoided.

So we think that policy languages and frameworks for the web should support both stateful and stateless protocols to face the variety of different needs of web applications.

4.5 What's new?

The existing approaches to trust management and trust negotiation already tackle the need for flexible, knowledge based interoperability, and take into account the main idiosyncrasies of the web—because ATN frameworks have been designed with exactly that scenario in mind. Today, to make a real contribution (even in the context of a policy-aware web), one should work on the open issues of trust management, that include at least the following topics:

– Negotiation success: how can we guarantee that negotiations succeed despite all the difficulties that may interfere? For example: rules not disclosed because of lack of trust; credentials not found because their repository is unknown. What kind of properties of the policy protection policy and of the *hints* (see Section 4.2) guarantee a successful termination when the policy "theoretically" permits access to a resource?
– Optimal negotiations: which strategies optimize information disclosure during negotiation? Do any reasonable preconditions prevent unnecessary information disclosure?
– A related problem is: In the presence of multiple ways of fulfilling a request, how should the client choose a response? One needs both a language for expressing preferences, and efficient algorithms for solving the corresponding optimization problem. While this negotiation step is more or less explicitly assumed by most works on trust negotiation, there is no concrete proposal so far.

Moreover, the integration of abductive semantics and ECA semantics is an open issue, as we have pointed out in a previous section. Of course this list is not exhaustive.

5 Cooperative policy enforcement

Cooperative enforcement involves both machine-to-machine and human-machine aspects. The former is handled by negotiation mechanisms: published policies, provisional actions, hints, and other metalevel information (see Section 4.2) can be interpreted by the client to identify automatically what information is needed to access a resource, and how to obtain that information. The human-machine interaction aspect deserves some discussion.

One of the causes of the enormous number of computer security violations on the Internet is the users' lack of technical expertise. In particular, users are typically not aware of the security policies applied by their system, not to speak about how those policies can be changed and how they might be improved by tailoring them to specific needs. As a consequence, most users ignore their computer's vulnerabilities and the corresponding countermeasures, so the system's protection facilities cannot be effectively exploited.

For example, it is well known that the default, generic policies that come with system installations—biased toward functionality rather than protection—are significantly less secure than a policy specialized to a specific context, but very few users know how to tune or replace the default policy. Moreover, users frequently do not understand what the policy is really checking up on, and hence they are unaware of the risks involved in many common operations.

Similar problems affect privacy protection. In trust negotiation, credential release policies are meant to achieve a satisfactory tradeoff between privacy and functionality (many interesting services cannot be obtained without releasing some information about the user). However, one cannot expect such techniques to be effective unless users are able to understand and possibly personalize the privacy policy enforced by their system.

Additionally, a better understanding of a web service's policy makes it easier for a first-time user to interact with the service. If a denied access results simply in a "*no*", then the user has no clue on how he or she can possibly acquire the permission to get the desired service (e.g., by completing a registration procedure, by supplying more credentials, by filling in some form, etc.) This is why we are advocating a form of *cooperative policy enforcement*, where negative responses are enriched with suggestions and other explanations whenever such information does not violate confidentiality (sometimes, part of the policy itself is sensitive).

For these reasons, WG I2 selected as one of its main objectives *greater user awareness and control on policies*. We are making policies easier to understand and formulate to the common user in the following ways:

- We adopt a *rule-based policy specification language*, because such languages are very flexible and at the same time structurally similar to the natural way in which policies are expressed by nontechnical users.
- We are making the policy specification language more friendly by developing a *controlled natural language* front-end to translate natural language text into executable rules (see next section).

– We are developing *advanced explanation mechanisms* to help the user understand what policies prescribe and control.

We have just published a deliverable on such explanation mechanisms [4]. It contains a requirements analysis for explanations in the context of automated trust negotiation (ATN). Moreover, we define explanation mechanisms for *why, why-not, how-to*, and *what-if* queries. There are several novel aspects in our approach:

– We adopt a *tabled explanation structure* as opposed to more traditional approaches based on single derivations or proof trees. The tabled approach makes it possible to describe infinite failures (which is essential for *why not* queries).
– Our explanations show simultaneously different possible proof attempts and allow users to see both local and global proof details at the same time. Such combination of local and global (intra-proof and inter-proof) information is expected to facilitate navigation across the explanation structures.
– We introduce suitable heuristics for focussing explanations by removing irrelevant parts of the proof attempts. Anyway, we provide a second level of explanations where all the missing details can be recovered, if desired.
– Our heuristics are *generic*, i.e. domain independent. This means that they require no manual configuration.
– The combination of tabling techniques and heuristics yields a completely novel method for explaining failure. In the past, the problem has been ignored or formulated differently.

Moreover, we make our explanation mechanisms *lightweight* and *scalable* in the sense that (i) they do not require any major effort when the general framework is instantiated in a specific application domain, and (ii) most of the computational effort can be delegated to the clients.

Queries are answered using the same policy specifications used for negotiation. Query answering is conceived for the following categories of users:

– Users who are trying to understand how to obtain access permissions;
– Users who are monitoring and verifying their own privacy policy;
– Policy managers who are verifying and monitoring their policies.

Currently, advanced queries comprise *why/why not, how-to*, and *what-if* queries.

Why/why not queries can be used by security managers to understand why some specific request has been accepted or rejected, which may be useful for debugging purposes. Moreover, why-not queries may help a user to understand what needs to be done in order to obtain the required permissions (a process that in general may include a combination of automated and manual actions). Such features are absolutely essential to enforce security requirements without discouraging users that try to connect to a web service for the first time. How-to queries have a similar role, and differ from why-not queries mainly because the former do not assume a previous query as a context, while the latter do.

What-if queries are hypothetical queries that allow to predict the behavior of a policy before credentials are actually searched for and before a request is actually submitted. What-if queries are good both for validation purposes and for helping users in obtaining permissions.

Among the technical challanges related to explanations, we mention:

– Finding the right tradeoff between explanation quality and the effort for instantiating the framework in new application domains. Second order explanation systems prescribe a sequence of expensive steps, including the creation of an independent domain knowledge base expressly for communicating with the user. This would be a serious obstacle to the applicability of the framework.

5.1 Natural language policies

Policies should be written by and understandable to the final users, to let them keep the behavior of their system under control. Otherwise the risk that users keep on adopting generic (hence ineffective) built-in policies, and remain unaware of which controls are actually made by the system would be extremely high—and this would significantly reduce the benefits of a flexible policy framework.

Of course, most users have no specific training in programming nor in formal logics. Fortunately, they spontaneously tend to formulate policies as rules; still, logical languages may be intimidating.

For this reason, we are designing front-ends based on graphical formalisms as well as *natural language interfaces*. We would like policy rules to be formulated like: "*Academic users can download the files in folder historical_data whenever their creation date precedes 1942*".

Clearly, the inherent ambiguity of natural language is incompatible with the precision needed by security and privacy specifications. For this reason we adopt a *controlled* fragment of English where a few simple rules determine a unique meaning for each sentence. This approach is complemented with a suitable interface that clarifies what the machine understands.

Some working group members have long standing expertise in controlled natural language, and a natural language front-end based on the ATTEMPTO system (`http://www.ifi.unizh.ch/attempto/`) is being progressively adapted to the need of policy languages, including negation as failure and deontic constructs.

6 Conclusions and perspectives

In our vision policies are really knowledge bases: a single body of declarative rules used in many possible ways, e.g., for negotiations, query answering, and other forms of system behavior control.

As far as trust negotiation is concerned, transparent interoperation based on ontology sharing can potentially become "everyday technology" in a short time (cf. Section 4.1). As such, trust negotiation may become a success story for semantic web ideas and techniques.

We are currently implementing the features described in this paper by extending the PeerTrust automated negotiation system (`http://www.learninglab.de/english/projects/peertrust.html`) and the parser of the Attempto system. Our current prototypes support real credential verification and real distributed computations, as well as the grammar structure and the anaphora mechanism underlying the sample natural language rule illustrated above.

Recently, the Semantic Web community declared interest in a notion of policy very similar to REWERSE's approach (see the paper co-authored by Hendler and Berners-Lee in these proceedings). While the examples outlined there are just special cases of what can be done in the old framework introduced in [5], there is evidence of interest in more advanced and powerful developments, in line with the curret goals of the research in ATN. There seems to be an emphasis on stateless interactions (i.e. degenerate negotiations consisting of one step only) defended by scalability arguments. However, we do not believe in such a priori restrictions. After all, the web started as a stateless protocol, but soon a number of techniques have been implemented to simulate stateful protocols and transactions in a number of real world applications.

We insist on the importance of *cooperative policy enforcement and trust management*, that give common users better understanding and control on the policies that govern their systems and the services they interact with. The closer we get to this objective, the higher the impact of our techniques and ideas will be.

References

1. M. Blaze, J. Feigenbaum, and M. Strauss. Compliance Checking in the PolicyMaker Trust Management System. In *Financial Cryptography*, British West Indies, February 1998.
2. P.A. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *IEEE 6th Intl. Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, pages 14–23. IEEE Computer Soc., 2005.
3. P.A. Bonatti and D. Olmedilla. Policy specification language. Technical Report I2-D2, REWERSE, Feb 2005. http://www.rewerse.net.
4. P.A. Bonatti, D. Olmedilla, and J. Peer. Advanced policy queries. Technical Report I2-D4, REWERSE, Aug 2005. http://www.rewerse.net.
5. P.A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002. Short version in the Proc. of the Conference on Computer and Communications Security (CCS'00), Athens, 2000.
6. Rita Gavriloaie, Wolfgang Nejdl, Daniel Olmedilla, Kent E. Seamons, and Marianne Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *1st European Semantic Web Symposium (ESWS 2004)*, volume 3053 of *Lecture Notes in Computer Science*, pages 342–356, Heraklion, Crete, Greece, may 2004. Springer.
7. N. Li, W. Winsborough, and J.C. Mitchell. Distributed Credential Chain Discovery in Trust Management (Extended Abstract). In *ACM Conference on Computer and Communications Security*, Philadelphia, Pennsylvania, November 2001.

8. Arnon Rosenthal and Marianne Winslett. Security of shared data in large systems: State of the art and research directions. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 962–964. ACM, 2004.
9. K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobsen, H. Mills, and L. Yu. Requirements for Policy Languages for Trust Negotiation. In *3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, CA, June 2002.
10. V. S. Subrahmanian, Piero A. Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus, Fatma Ozcan, and Robert Ross. *Heterogenous Active Agents*. MIT Press, 2000.
11. V.S. Subrahmanian, S. Adali, A. Brink, R. Emery, J.J. Lu, A. Rajput, T.J. Rogers, R. Ross, and C. Ward. Hermes: Heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/publications/ abstracts/hermes. html.
12. W. Winsborough, K. Seamons, and V. Jones. Negotiating Disclosure of Sensitive Credentials. In *Second Conference on Security in Communication Networks*, Amalfi, Italy, September 1999.
13. W. Winsborough, K. Seamons, and V. Jones. Automated Trust Negotiation. In *DARPA Information Survivability Conference and Exposition*, Hilton Head Island, SC, January 2000.
14. Marianne Winslett, Ting Yu, Kent E. Seamons, Adam Hess, Jared Jacobson, Ryan Jarvis, Bryan Smith, and Lina Yu. Negotiating trust on the web. *IEEE Internet Computing*, 6(6):30–37, 2002.
15. Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Trans. Inf. Syst. Secur.*, 6(1):1–42, 2003.
16. C. Zhang, P.A. Bonatti, and M. Winslett. Peeraccess: A logic for distributed authorization. In *12th ACM Conference on Computer and Communication Security (CCS 2005)*. ACM. To appear.