

Advanced Semantic Web Policies: Evolution, Reactivity, and Priorities

Philipp Kärger

L3S Research Center and Leibniz University Hannover, Germany

Abstract. Semantic Web policies are statements that define the behavior of a system acting on the Semantic Web. It turned out that current policy frameworks lack important features in both, their representation and their reasoning facilities. They typically do not allow for dynamics such as *evolution*, i.e., the consistent modification of the policy while it is evaluated at the same time, and *reactivity*, i.e., the ability to define events and actions in a policy. Policy frameworks are also not flexible enough to allow for *priorities*, i.e., a notion of weaker statements like “if possible do this, of not, do that”. In this paper, I describe why these features are crucial for advanced Semantic Web policies. Based on this, I motivate my PhD which aims at advanced Semantic Web policies providing both, dynamics including evolution and reactivity, and the flexibility to express priorities.

1 Research Topic Overview

In an open environment such as the Semantic Web, agents act on behalf of real users because systems are more efficient and less error-prone. Semantic Web agents automatically take decisions, react to events, and perform actions. For the definition of an agent’s behavior so-called policies are used describing how a system behaves under certain conditions. Semantic Web policies have recently gained a lot of research focus (see [1] for an overview). Among the diverse approaches to define a Semantic Web policy, rule-based policies play an important role. This is due to the fact that rules—being part of the Semantic Web stack—allow for a declarative definition of a system’s behavior. Moreover, there are powerful reasoning mechanisms available (such as logic programming) in order to evaluate rules and they are therefore exploitable for an efficient behavior control. I will focus on rule-based policy languages throughout this paper. With the help of the following scenario I am going to extract requirements an advanced Semantic Web policy framework should address.

A Semantic Web Scenario

A Semantic Web Broker is programmed with a set of declarative policy rules in order to notify members of a project about changes and progress in the project. Therefore the Broker collects information about the project exposed in the Semantic Web and passes it to registered and authorized members (some information is for project leaders only, etc.). Whether a user is registered or not depends

on several conditions, such as being an employee of a project partner, being an owner of a certain ID card, or having an entry in one's FOAF description. A user can prove her membership by providing digitally signed credentials. There are policies stating which member is notified about what kind of information update. In addition, project members themselves are allowed to define their preferences concerning when and how the Broker should notify them. Alice is a project member. One of her preferences states that she prefers to be notified via Instant Messenger. Only if her Messenger client is not available for any reason (e.g., because she is offline) she accepts notifications via email. Moreover, concerning the disclosure of credentials (e.g., in order to prove her project membership), she prefers to disclose her University Member Credential instead of her passport.

Requirements

Beyond the well-known requirements for Semantic Web policies such as expressiveness, simplicity, enforceability, scalability, and analyzability [2], three additional requirements can be extracted from the scenario:

1. Evolution. Systems acting on the Semantic Web as well as the Semantic Web itself are evolving. That is, systems change their behavior as well as the knowledge available in the Semantic Web changes. For example, the knowledge about whether a person is a member of the project or not may evolve. Also the policy specifying who is going to be notified about what may evolve. But what if during an ongoing transaction these policies change? What if the Semantic Web knowledge forming the base for a decision changes? These situations may cause inconsistent states of knowledge which have to be foreseen and avoided. This is of particular importance for policy-based trust negotiations [3] (e.g., for disclosing a membership credential in the scenario): trust negotiations may take longer than only one second and changing the policies in such cases may lead to an undesired disclosure of information [4]. In summary, policy frameworks should show consistent behaviour in case information changes during a transaction.

2. Reactivity. A Semantic Web system (as the Broker described in the scenario) has to be able to react. For example, the Broker is triggered whenever information about the project changes. That means, the Broker's policy states that, in case such a change is detected, a certain action has to be executed (in our case, project members will be notified). Therefore, a Semantic Web policy has to be *reactive* in its nature. This includes, on the one hand, a notion of events, composed events, and time as part of the policy language. On the other hand, reactions to events have to be enforced; therefore, actions, their sequences, and their flow control have to be part of the policy language's semantics.

3. Priorities. Policies are used to provide a system with a set of statements that exactly define its behavior. But, more often than not, reality is not as exact as a declarative rule. Some rules may contain a priority statement and therefore extend the meaning of the policy. For example, Alice prefers notifications via Instant Messenger to notifications via email. This forms a special kind of a policy: it is not clearly stated for which cases which alternative applies; it is solely stated that the first alternative should happen *if possible*, and which alternatives

she accepts if not. Therefore, it is needed to provide a policy language with subjective expressions such as "if possible, conclude this, if not, conclude that". These priority expressions introduce the possibility for more intuitive policy definitions.

Research Problem and Expected Impact

It turned out that current policy frameworks are far too restrictive to allow for evolution, reactivity, and notions of priorities. This forms the motivation of my PhD research: while being in my second year of PhD, I am concerned with studying how one can allow for these three requirements in a policy-based Semantic Web environment. As part of this, I study current Semantic Web policy frameworks, ongoing research towards Evolution and Reactivity on the Semantic Web as well as in logic programs, and logic programs allowing for preferences as part of their semantics. The goal of my PhD is to provide fundamental models for policy languages and policy frameworks that allow for these three basic requirements. I will develop a policy language that provides these requirements and implement a Semantic Web policy framework based on this language providing powerful policy reasoning mechanisms while still accounting for reactivity, evolution, and priorities.

2 Related Work and Challenges

In this section, I will focus on the three requirements extracted in the previous section, show that current Semantic Web policy frameworks do not address them, and where potential solutions have been studied already.

1. Evolution. Evolution on the Semantic Web has been studied intensively in recent years. The main focus of the research was notifying and distributing evolving knowledge in the Semantic Web by means of reactive rules. Although evolving knowledge has to be taken into account carefully (see next paragraph about reactivity), but what has not been addressed so far is the need for *evolving rules* on the Semantic Web: what if an agent's policies change while it is already acting in this very moment? What if the policy protecting certain private information changes during a policy-based negotiation? A first solution for that problem is provided in [4] in the area of policy-driven trust negotiations. Still, this approach does not completely address the problem of informing the negotiation partner about changes in the local policy. In [5] the evolution of digitally signed credentials are considered in the process of policy-based access control. The work in [5] defines several levels of consistency avoiding that a provided credential that becomes invalidated before the end of a trust negotiation causes an undesired disclosure of private information. But the authors left open which kind of inconsistencies may occur if during a negotiation *the policies themselves* become invalid, i.e., some policy statements are deleted, added, or modified. The evolution of rules in terms of updating a logic program is a hot topic in the area

of logic programming. The most prominent example is Dynamic Logic Programming [6]. This paradigm provides a semantics for the update of not only simple facts but also rules in generalized logic programs. Therefore, this paradigm fits well cases where the update of a logic program requires a possible revision of former knowledge. But the problem I am targeting at goes even further: it requires a semantics for updates which are performed *during* the evaluation of a logic program (in our case a set of policy rules). Therefore, Dynamic Logic Programming may provide a first step but does not handle updates at the granularity required for the updates of policies for example during a policy negotiation.

2. Reactivity. Reactivity as part of a powerful policy language has not been introduced so far. Except Ponder [7] which is an object oriented policy language that—even though it includes reactivity—does not support most of the other Semantic Web requirements (e.g., interoperability, well-defined semantics, reasoning, etc.). A reactive policy language requires a clear semantics for reactivity. Reactivity is a paradigm known from active databases where reactive rules were used to consistently update a database automatically according to modification of entries [8]. Later, reactivity became a prominent research field in the area of Semantic Web [9]. Reactivity on the Semantic Web is concerned with the reaction to certain events, that is, in most of the cases, updates of Semantic Web knowledge exposed in form of RDF statements. Reactions to events may include the update of another Semantic Web peer’s knowledge in order to keep a consistent knowledge in the whole Semantic Web. Typically, reactive behavior is described via Event Condition Action Rules (ECA-Rules) written in the form **ON Event IF Condition DO Action**. Current reactive Semantic Web solutions are, to mention a few, XChange [10], Reaction RuleML [11], and r^3 [12]. Solutions for my work will be inspired by those approaches. But, looking back at the scenario, it becomes clear that although ECA-frameworks provide reactivity, they are general-purpose Semantic Web reasoners for reactive rules and lack specific solutions for advanced agent control: they do not model specific interactions between agents which are needed for negotiations, delegations, and contextual information disclosure. Moreover, the evolution of rules is also not part of any ECA-language’s semantics.

3. Priorities and Preferences. Priorities or preferences are currently not part of any Semantic Web policy language. The definition of preference statements expressing priorities as part of a logic program has been addressed since one decade now. The idea behind such a statement is to express a human’s wishes of the form “**if possible conclude A if not conclude B if both are not possible conclude C ...**”. A solution for including these preference rules in logic programs was presented in [13]. Based on this work, a preference-enabled solution for policy-driven network control was introduced in [14]. Since this approach addresses the rather technical problem of network control, it does not include main requirements for a Semantic Web scenario, for example policy-driven negotiations and shared concepts. For more related work in that direction I point the

reader to [15]. In the database area, preferences are commonly used to allow for more intuitive and therefore user-friendly queries [16, 17]. These queries contain an “if possible” combined with an order of alternative values with decreasing preference. The result set of such queries is the well defined set of optimal results (often referred to as skyline of the preference query). This approach I extended for the use in policy-driven trust negotiations in [15] where preferences over credentials to be disclosed are exploited to guide a negotiation. Still, an advanced policy language should allow preferences on arbitrary predicates in the language, not only over credentials. Both orthogonal approaches, i.e., priorities in logic programs and preference queries in databases, provide some basic ideas of how preferences in policies may be formulated and evaluated; but they do not form a solution that incorporates priorities *into* an advanced policy language.

3 Achievements, Current State, and Work Plan

In the past first year of my PhD I was concerned with studying existing Semantic Web policy frameworks, preferences and evolution in logic programs, and reactivity on the Semantic Web. I was involved in the REVERSE project and took part in the development of Protune (PROvisional TRust NEgotiation), an expressive logic-based policy framework [18]. I studied preference-based approaches and developed a general theoretical model of how to incorporate preferences in policy-driven agent negotiations [15]. This work already forms a basis for a priorities-enabled policy framework. Together with the Computer Science Department of the Universidade Nova de Lisbon, Portugal, I am currently developing a reactive policy framework prototype based on r^3 and Protune. r^3 (Resourceful Reactive Rules) is a Semantic Web framework to express reactivity by means of ECA rules [12]. I also considered different use-cases and application scenarios for advanced policies (e.g., [19]). One particular direction I am following is to exploit advanced policies for extending current Instant Messenger and VoIP applications in order to allow for automated behavior control (as it is partially described in the scenario). Future steps of my PhD include the evaluation of advanced policies against these use-cases and application scenarios. The evaluations will be carried out in some of the research projects I am currently involved in such as OKKAM and TENCompetence [19]. Other steps of my PhD will focus on the remaining parts towards an advanced Semantic Web policy framework, namely to investigate current algebras of time in order to develop a suitable and powerful notion of time for a fine-grained definition of events in advanced policies. I plan to develop a model describing whether the modification of a rule set influences an ongoing decision process (i.e., reasoning process) thus solving the policy evolution problem. This will result in a policy negotiation model that includes notifications about such rule modifications. It is a known problem that preferences in logic programs increase complexity. Therefore, it has to be investigated which kind of preferences provide a sufficient expressiveness while being scalable at the same time.

References

1. Bonatti, P.A., Duma, C., Fuchs, N., Nejdil, W., Olmedilla, D., Peer, J., Shahmehri, N.: Semantic web policies - a discussion of requirements and research issues. In: ESWC, Budva, Montenegro, Springer (2006)
2. Tonti, G., Bradshaw, J.M., Jeffers, R., Montanari, R., Suri, N., Uszok, A.: Semantic web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. In: International Semantic Web Conference. (2003) 419–437
3. Winslett, M.: An introduction to trust negotiation. In: iTrust. (2003) 275–283
4. Skogsrud, H., Benatallah, B., Casati, F.: Trust-serv: model-driven lifecycle management of trust negotiation policies for web services. In: WWW. (2004)
5. Lee, A.J., Winslett, M.: Safety and consistency in policy-based authorization systems. In: 13th ACM conference on Computer and communications security. (2006)
6. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic updates of non-monotonic knowledge bases. *J. Log. Program.* **45**(1-3) (2000) 43–70
7. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The ponder policy specification language. In: POLICY 2001: Proceedings of the International Workshop on Policies for Distributed Systems and Networks, London, UK, Springer-Verlag (2001) 18–38
8. Widom, J., Ceri, S., eds.: Active Database Systems: Triggers and Rules For Advanced Database Processing. Morgan Kaufmann (1996)
9. Alferes, J.J., May, W.: Evolution and reactivity for the web. In: Reasoning Web. (2005) 134–172
10. Bailey, J., Bry, F., Eckert, M., Patranjan, P.L.: Flavours of xchange, a rule-based reactive language for the (semantic) web. In: RuleML. (2005) 187–192
11. Paschke, A., Kozlenkov, A., Boley, H., Tabet, S., Kifer, M., Dean, M.: Reaction RuleML. RuleML Initiative, <http://ibis.in.tum.de/research/ReactionRuleML/>. (2007)
12. May, W., Alferes, J.J., Amador, R.: Active rules in the semantic web: Dealing with language heterogeneity. In: RuleML. (2005) 30–44
13. Brewka, G.: Logic programming with ordered disjunction. In Kaufmann, M., ed.: In Proc. 18th National Conference on Artificial Intelligence, AAAI-2002. (2002)
14. Bertino, E., Mileo, A., Provetti, A.: PDL with preferences. In: Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05), Los Alamitos, CA, USA, IEEE Computer Society (2005) 213–222
15. Kärger, P., Olmedilla, D., Balke, W.T.: Using preferences for credential disclosure in policy-driven trust negotiations. In: 5th VLDB Workshop on Secure Data Management (SDM). (2008)
16. Chomicki, J.: Preference formulas in relational queries. *ACM Trans. Database Syst.* **28**(4) (2003) 427–466
17. Kießling, W.: Foundations of preferences in database systems. In: Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong, China (2002) 311–322
18. Bonatti, P., Olmedilla, D.: Driving and monitoring provisional trust negotiation with metapolicies. In: POLICY 2005: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, Washington, DC, USA, IEEE Computer Society (2005) 14–23
19. Coi, J.L.D., Kärger, P., Koesling, A.W., Olmedilla, D.: Exploiting policies in an open infrastructure for lifelong learning. In: 2nd European Conference on Technology Enhanced Learning (EC-TEL). Volume 4753 of Lecture Notes in Computer Science., Crete, Greece, Springer (Sep 2007) 26–40