A formal support for collaborative data sharing*

Fabio Martinelli, Ilaria Matteucci, Marinella Petrocchi, and Luca Wiegand

Istituto di Informatica e Telematica, CNR, Pisa, Italy
name.surname@iit.cnr.it

Abstract. Collaborating entities usually require the exchange of personal information for the achievement of a common goal, including enabling business transactions and the provisioning of critical services. A key issue affecting these interactions is the lack of control on how data is going to be used and processed by the entities that share it. To partially solve the issue, parties may have defined a set of data sharing policies regulating the exchange of data they own, or over which they have jurisdiction. However, distinct set of policies, defined by different authorities, may lead to conflicts once enacted, since, *e.g.,* different subjects may have defined different permissions on the same data set. This paper focuses on policy analysis and offers a formal support for coming up with a conflict-free set of data sharing policies. We illustrate the methodology on the example of an emergency management.

1 Introduction

An effective, speedy, and continuous data exchange is essential for todays life. In a collaborative fashion, several parties usually interact one with each other, for the achievement of a common goal. As an example, heating our houses is possible since a series of gas producers and gas distribution providers have agreed in cooperating to let the final product reach us. Such collaboration leads, with high probability, to a massive data exchange, that should be enabled in a safe way, avoiding the risks of violating privacy and confidentiality that may be associated with the data. In this scenario, it is of utmost importance to ensure that data exchange happens in accordance with well defined and automatically manageable policies. Data Sharing Agreements (DSA), which are formal agreements regulating how parties share data, enable secure, controlled, and collaborative data exchange. Consequently, infrastructures based on DSA become an increasingly important research topic and promise to be a flexible mechanism to ensure protection of critical data.

As the name itself recalls, a data sharing agreement is a contract signed by parties that mutually *agree* on its contents. According to the number of authors, we may distinguish between agreements with only one author, *unilateral* DSA, and agreements with more than one author, *multilateral* DSA.

* The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos) and under grant no 256980 (NESSoS), and from the IIT internal project Mobi-Care.

The core of a unilateral DSA consists of a list of rules regulating the sharing of information typically owned by the author of the contract. As an example, a unilateral DSA may dictate the set of privacy policies that an individual define on her own sensitive data, *e.g.,* medical or bank account data. To some extent, a contract edited by a service provider, and regulating the management of personal data of service consumer, could be considered as an unilateral DSA too. Indeed, accepting such a contract, service consumers implicitly agree on the data management policies dictated by the service provider. Whereas the client does not agree with the terms of the provider's data policies, she can always chooses another provider that best satisfies their privacy requirements. In [1], we presented a design phase for unilateral DSA, defining and developing two tools for DSA authoring and analysis.

On the other hand, a multilateral DSA consists of a document edited and signed by several parties. Each party has a set of privacy policies over a set of data. Data may be owned by the party itself, *e.g.,* the previous mentioned medical data of a patient, or the marketing strategical view of a company for the next five years. Also, some organization could have rights to express policies over data which it does not directly own, but over which it may have jurisdiction (*e.g.,* traffic policemen have usually rights to ask for driver licenses). Since each entity has its own rules regulating data sharing, and since data that are subjects of different policies may overlap, the design phase for multilateral DSA is quite more complex than the one for unilateral DSA. Drawing up a multilateral DSA requires, for instance, a definition phase in which policies dictated by different organizations over the same set of data are checked to be conflict-free.

In this paper, we extend the analysis framework presented in [1] to deal with multilateral DSA. We propose an analysis methodology, decorated by an analysis tool, as a formal support for the creation of a well-defined, conflict-free multilateral DSA. The analysis examples and results are presented through a set of reference policies related to a scenario in which a set of individuals/organizations need to share data in a urgent but controlled way, for the successful management of an emergency situation.

The paper is structured as follows. Section 2 describes the reference structure of a data sharing agreement. Section 3 shows the reference scenario. Sections 4 and 5 present our analysis framework. Section 6 discusses related work in the area. Finally, Section 7 concludes the paper.

2 Multilateral Data Sharing Agreements

From the analysis of samples of real DSA, *e.g.,* [2–4] we derived a general structure for an agreement. A DSA consists of various parts, among which a *Title*, a validity *Period*, the list of *Data* covered by the agreement, the list of involved *Subjects*, their respective *Signatures*, and *Data Sharing Policies* sections.

For the sake of classification, such policies can be divided into *Authorizations*, *Obligations*, and *Prohibitions*, indicating which actions are authorised, obliged, or denied on which data by which subject.

The following general assumptions hold throughout the paper:

1. if no obligation policies are explicitly expressed, then subjects are not required to make any actions on any data;
2. everything that is not explicitly expressed by an authorization or an obligation policy is prohibited.

From the second assumption, we can also derive that, if neither authorization nor obligation policies are explicitly expressed, then the following implicit prohibition holds: "all entities (users, groups, etc.) belonging to *Subjects* are not authorized/required to make any action on *Data* during *Period*".

Some sections in a DSA are optional: *Purpose* stating the purpose of the DSA in layman's language; *Definitions* defining terms used in the agreements; *Data quality* describing the degree of commitment to data quality; *Custodial responsibility* describing who is responsible for the data and the confidentiality requirements stated in the DSA; *Trust domain* defining the pre-existent trust relationships among the *Subjects*; and *Security infrastructure requirements* describing any requirement related to the security infrastructure, *e.g.,* : encryption algorithms, length of encryption keys, etc.

Hereafter, we suppose that a common ontology exists among the *Subjects* and we focus on the analysis of the authorizations, obligations, and prohibition sections.

3 Scenario

Let us consider an emergency scenario in which several vehicles are involved in an accident, including a tanker. Both firemen and Red Cross paramedics and toxicologists spring to the victims aid. We generically refer to firemen and Red Cross representatives as Rescue Time, or rescuers.

Managing the emergency with timeliness and accuracy implies to share information regarding the context in which rescuers operate. Examples of sensitive information the rescuers need to exchange are personal and medical information of victims, information on the tanker's content, and information on the alert state of the accident.

Reasonably, all the entities at stake, both individuals and organizations, have their own rules for sharing sensitive information, even within an emergency. Below, we list a series of plausible data sharing policies, in terms of authorizations A, obligations O, and prohibitions P.

Fire Brigade

A_{F1} Firemen can access both personal and medical data of the victim.
A_{F2} Firemen can access the personal data of drivers involved in an accident.
A_{F3} Firemen can access the delivery notes of any trucks involved in accidents. In particular, firemen can access the current delivery note and delivery notes of the past ten days.

A_{F4} Firemen can define the alert state of the accident.

P_{F1} If the alert state of the accident is greater than five, then rescue team members cannot communicate the alert state to the population living in the surrounding area.

Red Cross

A_{R1} Red Cross members (toxicologists plus paramedics) can access the alert state of the accident.

A_{R2} Paramedics can access medical data of the victim.

A_{R3} Toxicologists can access the delivery note of the trucks involved in the accident.

O_{R1} After that Red Cross members access the alert state of the accident, then, if the alert state is greater then five, then Red Cross members must communicate the alert state to the population living in the surrounding area.

Victim

P_{V1} People belonging to medical organizations cannot access my medical data if I am not in peril of my life.

Tankers company

P_{T1} Individuals not covering the role of firemen cannot access the current delivery notes of tankers.

4 Policy Specification

In order to specify information sharing policies, we adopt a controlled natural language called CNL4DSA [5]. The language aims at formally specifying such policies without loosing simplicity of use for end-users. Peculiarity of the language is the use of contexts, specifying attributes of *Subjects* and *Data* (like the subjects' roles, or the data category), plus attributes of environmental factors (like time and location). With the help of contexts, authorizations, obligations, and prohibitions are enriched with the capability to express under which set of conditions a subject is *allowed*, *obliged*, or *not allowed* to perform an action on a data object.

The core of CNL4DSA is the notion of *fragment*, a tuple $f = \langle s, a, o \rangle$ where s is the subject, a is the action, o is the object. The fragment expresses that "the subject s performs the action a on the object o", *e.g.,* "Bob reads Document1". It is possible to express authorizations, obligations, and prohibitions by adding the *can/must/cannot* constructs to the basic fragment. Fragments are evaluated within a specific *context*. In CNL4DSA, a *context* is a predicate c that evaluate either to *true* or *false*. Some examples of simple contexts are "date is more than 1 year ago" or "location is inside the building". In order to describe complex policies, contexts need to be composable. Hence, we use the Boolean connectors

and, *or*, and *not* for describing a *composite context* C which is defined inductively as follows:

$$C \ := c \mid C \ and \ C \mid C \ or \ C \mid not \ c$$

The syntax of a *composite fragment*, denoted as F, is inductively defined as follows:

$$F \ := nil \mid can/must/cannot \ f \mid F; F \mid if \ C \ then \ F \mid after \ f \ then \ F \mid (F)$$

The intuition is the following:

- *nil* can do nothing.
- *can*/*must*/*cannot* f is the atomic fragment that expresses that f is permitted/required/not permitted. $f = \langle s, a, o \rangle$. Its informal meaning is *the subject* s *can*/*must*/*cannot perform the action* a *on the object* o.
- $F; F$ is a list of composite fragments (*i.e.,* a list of authorizations, obligations, or prohibitions).
- *if* C *then* F expresses the logical implication between a context C and a composite fragment: if C holds, then F is permitted/required/not permitted.
- *after* f *then* F is a temporal sequence of fragments. Informally, after f has happened, then the composite fragment F is permitted/required/not permitted.

CNL4DSA has an operational semantics based on a modal transition system, able to express *admissible* and *necessary* requirements to the behaviour of the CNL4DSA specifications [5, 6].

4.1 Examples

With reference to the scenario in Section 3, we show some examples of CNL4DSA policies.

A_{F1} *if* hasRole(user1, fireman) *and* hasDataCategory(data, personal) *or* hasDataCategory(data, medical) *and* isReferredTo(data, user2) *and* isInvolvedIn(user2, accident) *then can* access(user1, data)

where hasRole(user1, fireman) *and* hasDataCategory(data, personal) *and* hasDataCategory(data, medical) *and* isReferredTo(data, user2) *and* isInvolvedIn(user2, accident) is a composite context and *can* access(user1, data) is a composite authorization fragment.

A_{F3} *if* hasRole(user1, fireman) *and* hasDataCategory(data, deliveryNote) *and* isReferredTo(data, truck) *and* isInvolvedIn (truck, accident) *then can* access(user1, data)

where hasRole(user1, fireman) *and* hasDataCategory(data, deliveryNote) *and* isReferredTo(data, truck) *and* isInvolvedIn (truck, accident) is a composite context and *can* access(user1, data) is a composite authorization fragment.

O_{R1} *if* hasRole(user1, RedCross) *and* hasDataCategory(data, alertState) *then after that* access(user1, data) *then if* isGreaterThan(alertState,five) *then must* communicate(user1,data)

where hasRole(user1, RedCross) *and* hasDataCategory(data, alertState) is a composite context, access(user1, data) is the simple fragment representing the pre-condition of the obligation, isGreaterThan(alertState,five) is a second composite context, and *must* communicate(user1,data) is a composite obligation fragment.

P_{T1} *if not* hasRole(user1,fireman) *and* hasDataCategory(data,deliveryNote) *and* isReferredTo(data,truck) *then cannot* access(user1, data)

where *not* hasRole(user1,fireman) *and* hasDataCategory(data,deliveryNote) *and* isReferredTo(data,truck) is a composite context and *cannot* access(user1, data) is a composite prohibition fragment.

5 Policy Analysis

In this section, we perform a series of analyses over a set of data sharing policies. The analysis process allows i) to detect conflict between policies; ii) to answer questions related to single clauses; and iii) to visualize a table of access.

- *Conflict detection.* After defining a set of contextual conditions, the analysis process is able to check if the set of policies is conflict-free, under those contextual conditions. Conflicts are searched either between an authorization and a prohibition clause, or between an obligation and a prohibition clause. Suppose that the user defines a certain context, *e.g.,* she defines the category of the data to be medical, and the role of the user to be a toxicologist. Looking for conflicts in the available policies means to check if

  1. there exists an authorization and a prohibition that, at the same time, allows and denies the toxicologist to perform the same action on those medical data;
  2. there exists an obligation and a prohibition that, at the same time, obliges and denies the toxicologist to perform the same action on those medical data.

  It is worth noticing that we do not check conflicts between an authorization and an obligation because our assumption is that any obliged action is implicitly authorized, see Section 2.

- *Questions related to single queries.* The analysis process is also able to answer several questions regarding authorizations, obligations, and prohibitions, like "is it true that subject x is authorised to perform action z on object y, under a set of contextual conditions?" and "is it true that subject x is required to perform action z on object y, after that subject w performs action t on object q, under a set of contextual conditions?". In the last sentence, possibly x=w, z=t, y=q.
- *Table of access.* This table shows all the authorised actions in the investigated set of policies, under a set of contextual conditions.

Answers to these questions are obtained with standard Maude built-in commands, such as *red*, *rew*, and *search* that basically allow to find all the possible traces, or a particular trace, in a specification written in Maude. The interested reader can refer to the Maude Manual, available online.

In [1] we verified that the policies of a unilateral DSA have been specified according to the author's intent. Hereafter, we exploit our analysis framework for detecting possible conflicts that can arise dealing with multiple policies to be deployed according to a multilateral DSA.

5.1 The Analysis Tool

The analysis tool consists of two parts:

- a formal engine that actually performs the analysis of the policies;
- a graphical user interface that allows the user to dynamically load contextual conditions and launch the analysis of the set of policies.

The Engine CNL4DSA has been designed with precise formal semantics rules, regulating states and transitions between these states. This allows for a precise translation of CNL4DSA in Maude. Maude is an executable programming language that models distributed systems and the actions within those systems [7]. Systems are specified by defining algebraic data types axiomatizing systems states, and rewrite rules declaring the relationships between the states and the transitions between them.

The choice of using Maude for DSA analysis is driven by the fact that rewrite rules are a natural way to model the behaviour of a distributed system, and we see a DSA exactly as a process where different subjects may interact with each other, possibly on the same set of objects. Maude is executable and comes with built-in commands allowing to search for allowed traces, *i.e.,* sequence of actions, of a policy specified in CNL4DSA. These traces represent the sequences of actions that are authorised, or required, or denied by the policy. Also, exploiting the implementation of modal logic over the CNL4DSA semantics, as done in [8, 9] for CCS-like languages, it is possible to prove that a modal formula, representing a certain query, is satisfied by the Maude specification of the DSA.

Also, a Maude related toolkit allows a series of formal reasoning about the specifications produced, including real-time and probabilistic model checking [10,

11]. These additional facilities allow to deal with DSA whose policies are also based on probability and time-out.

CNL4DSA has been made executable by translating its syntax and formal semantics in Maude and the translation has been presented in [1].

The Graphical User Interface The GUI is deployed as a Web Application and it allows the user to query the analysis engine and visualize its results. The analysis engine exposes its functionalities as Web Service methods. The GUI is in charge of retrieving the set of policies that a user wants to analyse and the related vocabulary. Each vocabulary is implemented as an ontology and the inner logic of the GUI exploits it in order to create and show a set of menus whose information is consistent with the vocabulary. We assume that all the organizations agree on a common ontology for expressing their policies. We leave the phase of negotiation of this common ontology as a future work.

The interface helps the user to create dynamic contexts, which represent the environment under which the analysis will be performed. The inner logic of the GUI updates the information according to the selected context. Furthermore, it is possible to compose different types of queries, related to authorizations, obligations, and prohibitions. Once the user selected the context and, possibly, the queries, the GUI sends all the inputs, *i.e.,* the vocabulary, the high level description of the policies, the context defining the conditions on which the policies have to be evaluated, and the set of queries to the engine that performs the analysis. When the analysis has been performed, the results are shown through the GUI.

5.2 Analysis Example

Here, we show some example analyses over our reference policies. The GUI is available at http://dev4.iit.cnr.it:8080/DsaAnalyzerWebGUI-0.1/?dsaID=CI.xml. The interested reader should press the *Submit for the Analysis* button in order to load our reference policies and the related vocabulary. The vocabulary is pre-loaded, *stitched* on the reference policies.

First, the user can select the contextual conditions under which the analysis is carried out. The user selects the context from a drop-down menu. The menu is dynamically created according to the vocabulary of the loaded policies. All the selected contexts are automatically set to *true*. We assume that everything that is not explicitly specified does not hold. Hence, the user shall select each context that is supposed to be true.

Once the context has been defined, the user has three possibilities (see Figure 2). Either she can ask for conflict detection, or she can compose a query and perform successive elaboration on it, or she can ask for the table of access.

In the following, we show how to compose a query and some analysis results when a conflict is detected.

Composition of queries. If the user selects the analysis *Compose a query*, a form appears through which it is possible to compose queries representing either authorizations, or obligations, or prohibitions, see Figure 3.

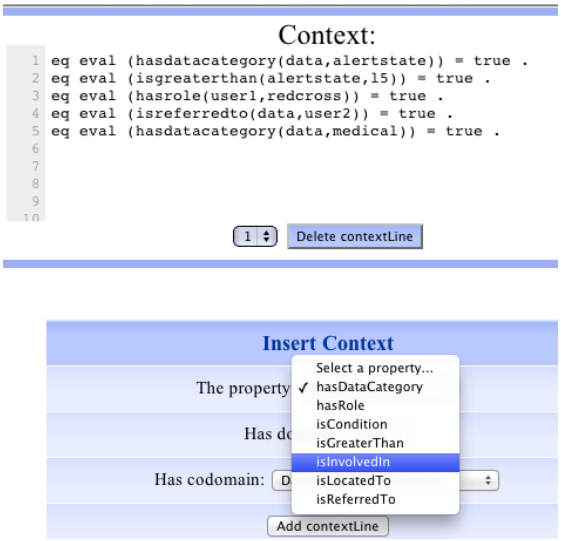

Fig. 1. Screenshot of the context insertion box.

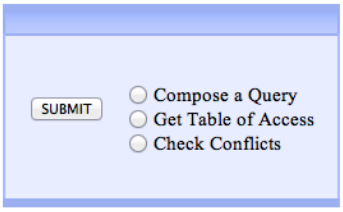

Fig. 2. Alternative analyses.

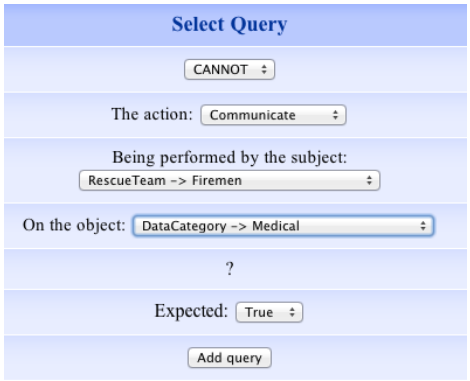

Fig. 3. Screenshot of the query insertion box.

Once that the user has selected both context and queries, she can start the analysis process by pressing the Submit button. This launches the inner analysis engine. At the end of the process, the GUI shows the analysis result to the user. In particular, the answer is true if a policy exists, among the loaded policy set, that satisfies the request represented by that query.

Conflict detection. We show some analysis examples where a conflict is detected between two data sharing policies defined by distinct organizations. The first conflict is detected between an authorization and a prohibition of our reference scenario.

Authorization A_{R2} defined by Red Cross and prohibition P_{V1} defined by an individual being involved in the accident lead to a conflict. Indeed, at the same time they give and deny to user1 the possibility to access the medical data of the individual. This happens when the following contextual conditions are set:

- data have data category *medical*
- user1 has role *paramedic*
- data are referred to *user2*
- *user2* is involved in *accident*

These conditions allow the paramedic to access the medical data (according to authorization A_{R2}). On the other hand, the individual is not in peril of her life. This is due to the fact that the context *user2 hascondition critical* is not true. The lack of this context activates prohibition P_{V1} according to which the paramedic is not allowed to access the data. The conflict detection is shown in Figure 4.

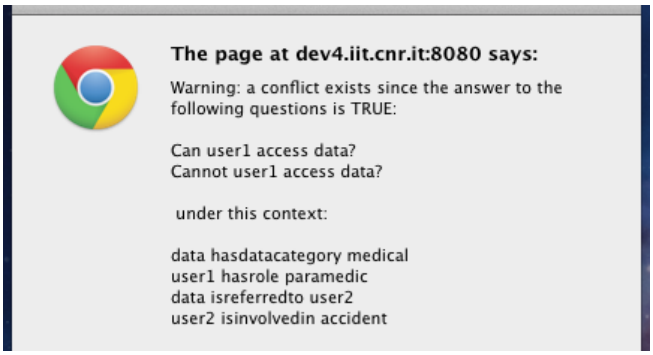

Fig. 4. Detection of conflict between an authorization and a prohibition

Authorization A_{R3} defined by Red Cross and prohibition P_{T1} defined by the tanker company lead to a conflict, see Figure 5. Indeed, at the same time they give and deny to user1 the possibility to access the truck delivery note. This happens when the following contextual conditions are set:

- user1 has role *toxicologist*

– data have data category *deliveryNote*
– data are referred to *truck*
– *truck* is involved in *accident*

Fig. 5. Detection of conflict between an authorization and a prohibition

Conflicts may also raise between obligations and prohibitions. Indeed, it is possible that some actions are prohibited by an organization and obliged by another one. This is the case of obligation O_{R1} by Red Cross and prohibition P_{F1} by Fire brigade. The Fire Brigade does not permit to communicate the alert state to people not belonging to the rescue team while Red Cross members are obliged to communicate the alert state, *e.g.,* to people living in the area surrounding the accident. The conflict detection raises with the following contextual conditions set to true:

– data have data category *alertState*
– *alertState* is greater than *level 5*
– user1 has role *RedCross*

Under this context, user1 is obliged to communicate the alert state, but, according to prohibition P_{F1} and since the Red Cross member is also a member of the rescue team, user1 cannot communicate the alert state. The conflict detection is shown in Figure 6.

Through the user interface it is possible to save the current configuration (*i.e.,* a set of contextual conditions and a set of queries) for successive elaborations (see Figure 7). This functionality allows to load a saved session without redefining contexts and queries. This is useful when the user, that possibly detects a conflict among the policies, modifies those policies. When checking the correctness of the modified clauses, there is no need to reformulate the contextual conditions and the queries.

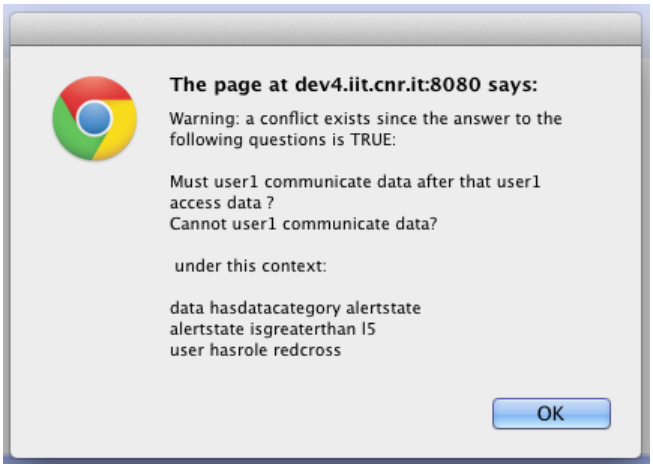

Fig. 6. Detection of conflicts between an obligation and a prohibition

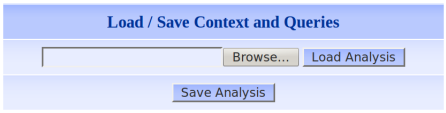

Fig. 7. Screenshot of the load and save box

Finally, the GUI is decorated with a help on line facility, for guiding the user through the capabilities of the analyser.

6 Related Work

Data protection in critical infrastructures has been discussed in the past recent years and several documents depict generic guidelines for secure data sharing in an informal way, *e.g.,* [12–15]. Often, such generic guidelines remain inaccessible from the software architecture supporting the data sharing itself, mainly because such guidelines are often written in natural language, which is difficult to parse and prone to ambiguity. Multilateral Data Sharing Agreements promise to be a flexible mean to fill the gap between a traditional legal contract regulating the sharing of data among different domains, and the software architecture supporting it. However, to come up with a consistent enforceable DSA, there is the need to check that data sharing policies deployed by different organizations/individuals are conflict-free.

The work presented in this paper mainly focuses on conflict detection among a set of data sharing policies originally defined by different authorities. In the literature, there exist other work related to the authoring, analysis, and enforcement of data sharing policies. Here, we revise our analysis framework with existing work in the area.

Binder [16] is an open logic-based security language that encodes security authorizations among components of communicating distributed systems. It has a notion for context and provides flexible low-level programming tools to express delegation, even if Binder does not directly implement higher-level security concepts like delegation itself. Also, the Rodin platform provides animation and model-checking toolset, for developing specifications based on the Event-B language (www.event-b.org). In [17], it is shown that the Event-B language can be used to model obliged events. This could be useful in the case of analysing obligations in DSA. In [18], the authors present a formalization of DSA clauses in Event-B and the ProB animator and model checker are exploited in order to verify that a system behaves according to its associated DSA. The main difference with our approach is that CNL4DSA captures the events (or actions) that a system can perform, the order in which they can be executed and it can be easily extended for dealing with other aspects of this execution, such as time and probabilities. On the other hand, in [18] the analysis of the agreement clauses is performed without considering a direct association between the set of clauses and the system functionality. Hence, Event-B language models the clauses that hold in a certain state of a system rather than its transition.

Also, a relevant work in [19] proposes a comprehensive framework for expressing highly complex privacy-related policies, featuring purposes and obligations. Also, a formal definition of conflicting permission assignments is given, together with efficient conflict-checking algorithms. Finally, the Policy Design Tool [20] offers a sophisticated way for modeling and analysing high-level security requirements in a business context and create security policy templates in a standard format.

To conclude, there exists generic formal approaches that could *a priori* be exploited for the analysis of some aspects of DSA. As an example, the Klaim family of process calculi [21] provides a high-level model for distributed systems, and, in particular, exploits a capability-based type system for programming and controlling access and usage of resources. Also, work in [22] considers policies that restrict the use and replication of information, *e.g.,* imposing that a certain information may only be used or copied a certain number of times. The analysis tool is a static analyser for a variant of Klaim.

Related to the sharing of data, but not strictly related to analysis, [23, 24] present on opportunistic authority evaluation scheme for sharing data in a secure way in a crisis management scenario. The main idea is to combine two already existing data sharing solutions in order to share data in a secure way through opportunistic networks. Finally, even if not specifically DSA-related, [25] presents a policy analysis framework which considers authorizations and obligations, giving useful diagnostic information.

7 Conclusions and Future Work

We focused on the analysis phase of a set of data sharing policies, originally defined by separate authorities for the management and the protection of data owned, or governed, by these authorities.

The achievement of a common goal, such as the management of an emergency, let such authorities interact and collaborate. Interactions may lead to the disclosure of possibly sensitive information whose sharing need to be regulated. Increasingly used, data sharing agreements are a usual way to regulate the sharing of information. In this paper, we propose a formal analysis framework to support several authorities to come up with the definition of a conflict-free multilateral DSA. The framework consists of a user-friendly interface that exploits capabilities of a background analysis tool in such a way to guide the user to detect conflicts on a multilateral DSA.

We leave some work for the future. Currently, the vocabularies collecting the terms used in a DSA do not carry semantic information, but we plan to evolve them towards more formal ontological definition of terms in such a way to enable the management of different vocabularies in which syntactically different terms are semantically equivalent, *e.g.,* in which two different terms refer to the same subject, or object. Also, our tool is able to detect conflicts, but no strategy is being defined and enforced to solve them. We are currently working on a classification of different kind of conflicting policies and on the definition of a set of strategies for supporting the user in solving conflicts, once detected. Finally, as it is common for tools based on state exploration, the underlying analysis engine suffers from the problem of the state explosion. Thus, it may be convenient to further investigate the feasibility of using this engine for more complex DSA specifications.

References

1. Matteucci, I., Petrocchi, M., Sbodio, M.L., Wiegand, L.: A design phase for data sharing agreements. In: DPM/SETOP. (2011) 25–41
2. Oklahoma Health Care Authority: Interagency Agreement. www.okhca.org/provider/contracts/ffs/pdflib/mhcm_over21.pdf (Last access 11/04/2012)
3. National Research Network: Data Sharing Agreement Template. www.researchtoolkit.org/primer/docs/AAFP-NRN DUA.pdf (Last access 11/04/2012)
4. National Collaborative on Workforce and Disability: Sample Inter-Agency Data Sharing Agreement. http://www.ncwd-youth.info/assets/guides/assessment/sample_forms/data_share.pdf (Last access 11/04/2012)
5. Matteucci, I., Petrocchi, M., Sbodio, M.L.: CNL4DSA: a Controlled Natural Language for Data Sharing Agreements. In: SAC: Privacy on the Web Track, ACM (2010) 616–620
6. Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS. (1988) 203–210

7. Clavel, M., et al., eds.: All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic. In Clavel, M., et al., eds.: All About Maude. Volume 4350 of LNCS., Springer (2007)
8. Verdejo, A., Martí-Oliet, N.: Implementing CCS in Maude 2. ENTCS 71 (2002)
9. Colombo, M., Martinelli, F., Matteucci, I., Petrocchi, M.: Context-aware analysis of data sharing agreements. In: Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services. (2010)
10. Ölveczky, P.C., Meseguer, J.: Semantics and pragmatics of Real-Time Maude. Higher-Order and Symbolic Computation **20**(1-2) (2007) 161–196
11. AlTurki, M., Meseguer, J.: PVeStA: A parallel statistical model checking and quantitative analysis tool. In: Proceedings of CALCO11: The 4th International Conference on Algebra and Coalgebra in Computer Science, Winchester, UK. (2011)
12. North American Electronic Reliability Corporation: Critical infrastructure protection: Security guidelines. http://www.nerc.com (Last access 19/04/2012)
13. U.S. Department of Justice: Justice information sharing. http://it.ojp.gov/default.aspx (Last access 19/04/2012)
14. Natural Resources Canada: Best practices for sharing sensitive environmental geospatial data. www.geoconnections.org (2010)
15. US Fire Administration: Critical infrastructure protection – information sharing and analysis center. http://www.usfa.fema.gov/fireservice/subjects/emr-isac/ (Last access 19/04/2012)
16. Abadi, M.: Logic in Access Control. In: LICS, IEEE (2003) 228
17. Bicarregui, J., et al.: Towards Modelling Obligations in Event-B. In: ABZ. (2008) 181–194
18. Arenas, A., et al.: An Event-B Approach to Data Sharing Agreements. In: Integrated Formal Methods, Springer (2010) 28–42
19. Ni, Q., et al.: Privacy-aware Role-based Access Control. ACM Transactions on Information and System Security **13** (2010)
20. Policy Design Tool: http://www.alphaworks.ibm.com/tech/policydesigntool (2009)
21. De Nicola, R., Ferrari, G.L., Pugliese, R.: Programming Access Control: The KLAIM Experience. In: CONCUR. (2000) 48–65
22. Hansen, R.R., Nielson, F., Nielson, H.R., Probst, C.W.: Static Validation of Licence Conformance Policies. In: ARES. (2008) 1104–1111
23. Scalavino, E., Gowadia, V., Lupu, E.C.: PAES: Policy-Based Authority Evaluation Scheme. In: DBSec. (2009) 268–282
24. Scalavino, E., Russello, G., Ball, R., Gowadia, V., Lupu, E.C.: An Opportunistic Authority Evaluation Scheme for Data Security in Crisis Management Scenarios. In: ASIACCS. (2010)
25. Craven, R., et al.: Expressive Policy Analysis with Enhanced System Dynamicity. In: ASIACCS. (2009)