Specifying Data Sharing Agreements

Vipin Swarup, Len Seligman, and Arnon Rosenthal
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102
{swarup, seligman, arnie}@mitre.org

Abstract

When consumers build value-added services on top of data resources they do not control, they need to manage their information supply chains to ensure that their data suppliers produce and supply required data as needed. Producers also need to manage their information supply chains to ensure that their data is disseminated and protected appropriately. In this paper, we present a novel model for data sharing agreements that supports a wide variety of data sharing policies. The model is based on distributed temporal logic (DTL) predicates that are expressed over events in dataflow graphs. A dataflow graph's nodes are principals with local stores, and its edges are (typed) channels along which data flows. We illustrate the model via examples and discuss the kinds of analyses enabled by the model.

1 Introduction

Service Level Agreements (SLAs) are used by organizations to express obligations and expectations regarding service-level parameters. For instance, for network provider services, SLAs describe obligations over parameters such as availability, latency, throughput, packet loss, etc. Similarly, for data sharing services that focus on the sharing of information, SLAs can describe a variety of obligations over parameters such as the above.

However, there are several key aspects of data sharing services that are not addressed by traditional SLAs that focus on functional business services. First, data sharing obligations may require a provider to actively engage in actions that result in wider sharing of its data. These obligations may include parameters such as data freshness and quality, regular update dissemination, etc. Second, data sharing obligations may require a data recipient to further share the data, share derivatives of the data, or share audit records of actions that it invoked on the data. Third, obligations may restrict what the recipient may do with the data, e.g., whether the recipient can print a document. Finally, data objects subsume other data objects and this property can be exploited—e.g., an agreement about European Union persons' data is relevant to a demand for German persons' data.

Currently, data sharing obligations between large organizations are captured in text documents called Memoranda of Understanding (MOUs) or Memoranda of Agreement (MOAs). Current practice is to use textual memoranda, an approach with several disadvantages. First, there is little help for the writers, so important sharing issues are often omitted. Second, the documents are typically filed away in a drawer and seldom used thereafter. Third, it is hard to provide automated support for reasoning about the contents of textual memoranda.

In this paper, we present a model for *data sharing agreements* (DSAs) that encode data sharing obligations. We focus on obligations about data stores and data flows, although the model can be extended to express obligations about the collection and processing of data, the use of data by parties, etc. We represent data stores as collections of typed values, dataflows as data streams between principals, and obligations as temporal constraints on data store and data stream events. Our model was developed by studying several real-world Memoranda of Agreements (MOAs) that are used by large U.S. government organizations to capture data sharing obligations.

The full version of this paper [6] contains a specification language based on the model described here, as well as example specifications of DSAs that are motivated by real-world MOAs. It also contains a detailed comparison with related work and highlights how our model improves on previous approaches.

2 Data Sharing Agreements (DSAs)

DSAs share many aspects of SLAs, for instance, descriptions of the parties to the agreements, availability constraints, and temporal constraints like agreement lifetimes [3]. They also inherit from the general notion of

agreements, that a party may incur obligations in return for benefits. For example, a consumer may promise to pay cash, to refrain from certain activities (e.g., non-compete agreements), or even to supply the consumer's own information to competitors. However, the primary purpose of DSAs is to capture data sharing clauses including descriptions of the data being shared, and obligations that constrain both the providers and consumers of the data and the data flows among them. The data may be described using standard techniques such as relational schema, XML schemas or DTDs, or object classes and we do not elaborate on the data model here.

Obligations on data providers can concern both the need to send certain data (which the provider must then produce or acquire) and the quality of what is sent. They include recency constraints (data will reflect real world events, or data updates will be forwarded, with specified promptness or periodicity); visibility constraints (access will be provided to specified data views); and quality constraints (shared data will be of specified freshness, accuracy, precision, etc.).

Obligations on data consumers include usage controls that enforce data provider protection policies (e.g., data will not be copied, data will be deleted after 3 days, etc.); dissemination controls that enforce data provider sharing policies (e.g., original source will be credited, subsequent research products based on the data will be shared back with the data providers; providers will be notified of who has accessed the data); and security constraints (e.g., providers will be notified of security breaches in consumer networks). Note that parties may be data consumers in one DSA, and providers (e.g., of derived products) in another. Parties may even be both data consumers and providers in the same DSA (e.g., providers of logistical data and consumers of end-user audit data).

All obligations may be conditional on events (e.g., receipt of a data object, detection of attack or compromise, etc.) and state predicates (e.g., declaration of local emergency, system failures, relationship among data values). Further, data sharing agreements may include obligations whose fulfillment depends on other DSAs or SLAs. For instance, suppose that B is obliged to ensure that C receives fresh and accurate data at regular intervals. In order to meet this obligation, B might rely on DSAs with data suppliers and SLAs with network providers. Finally, parties to a DSA may agree to inherit obligations from other DSAs.

A DSA's obligations impose global constraints on access control policies of various parties, even in future states. The exact nature depends on how the parties meet the obligations. For instance, suppose that a data recipient B is obligated to share data updates back with the provider A. Then either A must be given access rights to the data, or B must allow release and create a process that pushes the update notifications to A.

3 DSA Model

A DSA is an agreement between a set of principals regarding the sharing of data among themselves. In this context, data sharing refers to the explicit flow of data from one principal to another, and not to the subtler notions of information flow or data inference. Hence, we model a DSA as a set of predicates expressed over a dataflow graph whose nodes are principals with local stores, and whose edges are (typed) channels along which data flows.

Principals can be specified in well-understood ways and include both simple principals (e.g., named individuals or organizations) and compound principals (e.g., groups, roles, etc.).

A DSA pertains to specific data schemas and data instance sets. Data may be represented in any *data model*, e.g., the relational data model, the XML data model, or the object data model.

Each principal is associated with a local data store. A *data store* is a function that maps (location) names to data values. The data values may be data relations, data streams, documents, objects, etc., depending on the underlying data model.

Data resources describe the data to which a DSA pertains. A data resource is a tuple $\langle rn, p, DV \rangle$, where rn is a globally unique data resource name, p is the principal offering the data resource, and DV (i.e., a data view) is a set of tuples $\langle q, T \rangle$, where q is a query language expression over p's local data store and T is the type of the query's result. A data resource might describe both the data content that is being shared, and metadata such as the sharing context and attributes of principals.

A *dataflow* [4] F is a tuple $\langle s, d, T \rangle$ where s and d are principals and T is a type. F represents a data stream of values of type T flowing from source s to destination d. The source s can place value o of type T into the dataflow stream (which we write as $F.send(o)$, while the destination principal d can read values from the stream ($F.receive(o)$). The state of a dataflow F is a tuple $state(F) = \langle f, r \rangle$ where $f \in T^*$ is the sequence of values that have been placed in the stream, and $r \in T^*$ is the sequence of values that F's destination principal has read from the stream. Data values can include message identifiers to capture which values in the stream have been received.

Obligations are expressed as formulae in Distributed Temporal Logic (DTL) [1, 2] which is a generalization of Linear Temporal Logic (LTL). Obligation formulae specify properties of traces of dataflow events (sending and receiving data) and data store events (updating data stores). DTL includes both past-time and future-time temporal operators including **Y** (*previous*), **P** (*sometime in the past*), **H** (*always in the past*), **S** (*since*), **X** (*next*), **G** (*always*), and **U** (*weak until*). A DTL formula can refer to a specific principal's lo-

```
e    ::=  c.src | c.dest | getloc(Q,x) | ...
p    ::=  c.send(e) | c.rcv(e) | setloc(Q,x,e) | ...
ψ    ::=  p | ψ and ψ | ψ or ψ | not ψ | ψ at Q |
          (forall τ x) ψ | (exists τ x) ψ | if p then ψ |
          ψ innext i | ψ until i | ψ until ψ |
          ψ atnext i | ψ inprev i | ψ fromprev i |
          ψ fromprev ψ | ψ atprev i | ( ψ )
```

Figure 1. DSA Obligation Language Snippet

cal data space and hence can be true only for that specific principal. Thus, the DTL formula $@_a[\psi]$ asserts that proposition ψ holds in the local context of principal a.

Finally, a *data sharing agreement* DSA is a tuple $\langle P, S, DS, DR, DF, O\rangle$ where P is a set of principals (i.e., the parties referenced in the agreement), $S \subseteq P$ is the set of signatories of the agreement, DS is a function that maps each principal in P to a data store (that represents the local data store of the principal), DR is a set of data resources that are views of the data stores in DS, DF is a set of dataflows between principals in P, and O is a set of obligations of principals in P. Note that only principals in S are signatories of the DSA and hence have agreed to satisfy those obligations that oblige them.

4 Examples

We now present several small examples of sharing obligations that can be expressed in our model. These examples are specified using our DSA language, a snippet of which is presented in Figure 1. The full version of this paper [6] describes our complete DSA specification language, and it includes a specification of a DSA based on a real MOA between large U.S. government organizations. For brevity, we have taken significant liberty with the syntax here, e.g., by omitting certain clauses such as cancellation and penalty clauses in obligations. Informally, c.src and c.dest are expressions that refer to the source and destination of channel c respectively, while getloc(Q,x) denotes the value bound to location x in Q's data store. c.send(o) and c.rcv(o) are predicates that hold if object o was just sent and received (respectively) on channel c, while setloc(Q,x,e) is a predicate that holds if Q's data store was just updated to bind location x to value e. We use syntactically sugared DTL operators to express temporal conditions, e.g., ϕ **innext** t specifies that ϕ must hold sometime in the next t timesteps. Finally, ϕ **at** B specifies that ϕ must hold in the local state of principal B.

In the examples below, let A, B, and C be principals; $c1$ be a channel (of type T) from A to B; $c2$ be a channel from B to C; and $c3$ be a channel from B to A.

Responsive forwarding: B will send C on channel $c2$ each object it receives from A on channel $c1$ within 24 hours of receiving it:

(**forall** T o) **if** c1.rcv(o) **then**
  (c2.send(o) **innext** 24 hrs
at B **until** 1/1/2007

Nondisclosure agreements: If B receives object o from A on channel c1, then B will not thereafter send o to any other principal for a year.

(**forall** T o) **if** c1.rcv(o) **then**
  (**forall** Channel c) **if** (c.dest $\neq$ A) **then**
    (**not** c.send(o) **until** 1 year)
at B **until** 1/1/2007

Usage notification: If B receives object o from A on channel c1, then for the next 365 days, B will notify A each time B sends o to another principal. We construct the notification object via an externally defined function called "notify".

(**forall** T o) **if** c1.rcv(o) **then**
  (**forall** Channel c)
    if (c.send(o) **and** c.dest $\neq$ A) **then**
      c3.send(notify(B,c.dest,o)) **innext** 1 day
  until 365 days
at B **until** 1/1/2007

Recurrence: A will send B the latest update to object o every 24 hours.

if c1.send(o) **then**
  c1.send(update(o)) **innext** 1 day
at A **until** 1/1/2007

Privacy: Privacy obligations may arise when a person or an organization shares personal data with another organization. They may also arise when an organization receives data about a human, imposed by government (via laws) rather than the provider.[1] The simple privacy obligation specified below asserts that A must delete all personal information about B from location x within one year of A's storing it.

(**forall** T e) **if** setloc(A,x,e)
    and (e.subject = B) **then**
  setloc(A,x,null) **innext** 1 year
at A **until** 1/1/2007

[1] Some obligations (e.g., those imposed by law) are implicit—they are not part of any explicit DSA even though the parties are obliged to satisfy them. Such implicit obligations may be specified in our model although they are not the focus of this paper.

The above examples can be combined to form complex obligations. For instance, if B receives data O from A, then B will not share O with foreign citizens, and will notify A about US citizens who are shown the data. Furthermore, even for US citizens, B will not reveal that A was the source. Our model is quite powerful and can capture a wide variety of such data sharing policies.

5 Analyses

We sketch the kinds of analyses that are enabled by a formal data sharing model. Tractability of some of these analyses requires a data model with finite data types and a tractable query language.

- What are my obligations to share my data with others? E.g., to whom am I obligated to share California customers' data?
- What are other people's obligations to share data with me? E.g., who is obligated to supply me with data about Massachusetts vendors?
- Which obligations will I be unable to satisfy? E.g., we had a tornado, and lost a houseful of medical data in Kansas. E.g., we're obliged to share all data with a business partner but we receive product documentation under a stringent nondisclosure agreement.
- Do I need a new DSA, or are my information needs already covered? What data can I rely on of the following sort? What new DSA do I need to sign to receive desired data?
- What actions must I take in order to meet all my obligations? Can we automatically generate dataflows from a DSA that satisfy the DSA's dataflow obligations?
- I am considering (or have decided on) a change to the data schema or to the frequency of data updates. Which of my consumers are affected and what are my obligations to them?

6 Conclusion

In this paper, we have presented a novel model for data sharing agreements. The model is based on a dataflow graph whose nodes are principals with local stores, and whose edges are (typed) channels along which data flows. Data sharing constraints are expressed as DTL predicates over data stores and data flows. These constraints can include both *past* events and *future* events, and may hold only at certain principals' local states. To the best of our knowledge, this is the first such model of data sharing agreements.

Our work has been motivated by several real-world data sharing agreements (currently expressed as textual Memoranda of Agreement). Our full paper presents a DSA specification language and illustrates the power of our language by expressing a sanitized fragment of a real MOA. Since our language can be given a precise formal semantics, DSAs specified in our language may be subject to a variety of automated analyses. Tractability of these analyses depends on the properties of the underlying data model and DTL.

This work is an important first step towards our ultimate goal of building a comprehensive technical infrastructure for data sharing agreements [5]. We are beginning to develop a prototype platform that manages and enforces DSAs. This platform will include wizards to assist in the creation of comprehensive and consistent DSAs; repositories to assist in the lifecycle management of DSAs; agents to monitor and enforce the terms of the DSAs, when possible; and modules to provide automated analysis capabilities.

Note that obligations are binding on the obliged parties and the parties may be subject to penalties for failing to meet their obligations. Hence, obligation systems are subject to a variety of attacks by adversaries. Attacks include creating obligations that are not undertaken by the obliged principals, freeing principals from their obligations, causing principals to violate their obligations, and preventing obligation monitors from detecting violations of obligations. Hence, a secure DSA system must protect against such attacks while managing DSAs and monitoring for potential violations of obligations.

References

[1] H.-D. Ehrich and C. Caleiro. Specifying communication in distributed information systems. *Acta Inf.*, 36(8):591–616, 2000.

[2] M. Hilty, D. Basin, and A. Pretschner. On obligations. In *10th European Symposium on Research in Computer Security (ESORICS 2005)*, volume 3679 of *Lecture Notes in Computer Science*, pages 98–117. Springer, 2005.

[3] A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management, Special Issue on E-Business Management*, 11(1), March 2003.

[4] G. T. Leavens, T. Wahls, and A. L. Baker. Formal semantics for SA style data flow diagram specification languages. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 526–532, 1999.

[5] L. Seligman, A. Rosenthal, and J. Caverlee. Data service agreements: Toward a data supply chain. In *Workshop on Information Integration on the Web, at VLDB 2004*, 2004.

[6] V. Swarup, L. Seligman, and A. Rosenthal. Specifying data sharing agreements. MITRE Paper MP 06W0000065, The MITRE Corporation, McLean, VA 22101, 2006.