An Automated Framework for Supporting Data-Governance Rule Compliance in Decentralized MIMO Contexts

Rui Zhao
University of Edinburgh
rui.zhao@ed.ac.uk

Abstract

We propose Dr.Aid, a logic-based AI framework for automated compliance checking of data governance rules over data-flow graphs. The rules are modelled using a formal language based on situation calculus and are suitable for decentralized contexts with multi-input-multi-output (MIMO) processes. Dr.Aid models *data rules* and *flow rules* and checks compliance by reasoning about the propagation, combination, modification and application of data rules over the data flow graphs. Our approach is driven and evaluated by real-world datasets using provenance graphs from data-intensive research.

1 Introduction

The significant accomplishments of Artificial Intelligence and Data Science in the recent years have highlighted the value and importance of combining data sets from multiple sources, particularly in areas of global interest, such as healthcare. As we deal with such large collections of personal, sensitive, or otherwise valuable data, issues of data privacy, ownership and governance gain dominance.

Data sharing is typically bound by data-governance rules or data-use policies (*policies* for short) established by the data provider and adopted by data users. However, compliance checking is usually performed manually, in a time consuming and error prone way. This diminishes the trust of the data provider in the users' ability to comply with the agreed rules. Coupled with increasingly strict legislation for responsible data sharing, data providers are reluctant to release any data with the slightest risk of public objections. This has created a polarization in data-governance rules: data is either released freely as *open data* or under strict barriers to access, such as restricted data safe havens with controlled access, strict governance rules, meticulous and tedious application processes, supervision, etc. This has a direct impact on establishing and scaling collaborations for AI and data-intensive research across organisations and jurisdictions.

We propose the automation of compliance checking of policies using a symbolic AI agent. The agent facilitates enforcement of the desired policies, providing peace of mind to both data providers and users. It can also provide broader support, for instance by automatically deriving policies associated with data products and providing reminders for required actions (such as reporting data use at specific milestones).

More specifically, we envision a logic-based formalisation of data-governance rules coupled with an automated reasoning mechanism that can track and derive policies through every stage of the data-use workflow. This includes propagating data-governance rules from one or more data inputs to the corresponding outputs accordingly, merging policies from different upstreams, modifying the policies to reflect changes in the underlying data, and checking the application of policies in the current processes.

With this goal, we have developed a formal language based on situation calculus to model data-governance rules and their propagation and a corresponding framework, *Dr.Aid* (Data Rule Aid), to perform compliance reasoning on data-flow graphs via logic-based querying. We are evaluating our approach using real-world data-provenance graphs (a standardised representation of actions on data) from cyclone tracking and computational seismology applications.

Prior research has been developed under restricted assumptions, such as having a single context, a single stage of processing, propagating rules unchanged without reflecting modifications in the data [Elnikety *et al.*, 2016], or assuming linear processing [Pasquier *et al.*, 2017]. Dr.Aid addresses all of these issues as they arise in practice. It is explicitly designed for MIMO processes and multi-staged data flows, and dynamically updates rules to reflect data modifications.

2 Approach

Our framework, Dr.Aid, checks compliance with data-governance rules (policies) suitable for decentralized MIMO data processing, using a formal language and semantics. It enables the modelling of data-use policies for decentralized MIMO contexts, and performs logical reasoning to deliver information about policies for specific workflow runs.

Our formal rule language consists of two inter-operating parts: *data rules* and *flow rules* [Zhao and Atkinson, 2019]. *Data rules* encode data-governance rules for multi-staged processing, e.g. "*users must properly acknowledge the data providers*"; *flow rules* represent the changes of data rules in each process as a result of data transportation and transformation, e.g. "*column 3 from input 1 is changed to column 2 on output 2*". Thus, the flow rules specify how data rules are propagated and transformed from inputs to outputs for each

process; data rules are propagated between processes by following the data-flow graph. The reasoner combines these two forms of rule handling for each workflow enactment.

The two main elements of our data rules are *attributes*, each of which is a triple (N, T, V) of a name N, a type T and a value V describing properties of the data, and *obligations*, each of which is a triple (OD, VB, AC) consisting of an obligation definition OD (the obligated action to perform upon activation), a validity binding VB (describing additional applicability constraints), and an activation condition AC (the triggering condition). The attribute type T and the obligated action type of OD can refer to external definitions through a semantic approach, thus allowing the rule to unambiguously interoperate across institutional boundaries.

Our formalisation uses situation calculus to logically describe each stage (situation) of the workflow. For example, attributes (and similarly obligations) are attached to a list H of history they have been through (for disambiguation) and a situation S that they apply to, in the fluent $attribute(N, T, V, H, S)$. The original data rules are fluents that hold in the initial situation $s0$.

For instance, a *data rule* may dictate that field #3 in the data is *private* and any use of it should be reported to the data provider. This would involve (a) an attribute which we name pf (private field), a type $column$ describing the fact that it refers to a specific field, and a value 3 to indicate the 3rd field. and (b) an obligation requiring a $report$ action be taken when *any* action is performed ($action = *$) on the bound data:

$$attribute(pf, column, 3, [input1, pf\_1], s0).$$
$$obligation(report, [[input1, pf\_1]], action = *, input1, s0).$$

The *flow rules* describe how data rules flow through the process, by propagating and manipulating data rules in order to reflect how modifications performed in the data affect the policies. For example, if a process changes the column order in one of its outputs, then the column index in the data rule is changed for that output for downstream processes. This modification is encoded as a flow rule as follows:

$$edit(input1, output2, *, column, 3, column, 2)$$

This states that the process changes *column 3* to *column 2* for data coming in from port $input1$ and results to port $output2$, matching attributes with *any* ($*$) name. The mechanism to explicitly refer to input and output ports for each process is what allows us to formulate flow rules in a MIMO setting.

The Do function applies a series of flow rules to a situation. As an example, let us consider the following situation:

$$\begin{aligned} s1 = Do(&pr(input1, [output1, output2]) : \\ &edit(input1, output2, *, column, 3, column, 2) : \\ &end([output1, output2]), s0)) \end{aligned}$$

In this, $s1$ is the result of a *propagation* rule ($pr(P_{in}, Ps_{out})$ propagates all data rules from port P_{in} to every port in Ps_{out}), the $edit$ rule discussed above, and the end of the process with 2 outputs (one untouched and one edited). Querying is then analogous to the projection task, i.e. querying fluents that hold at the targeted final situation. For example, the query $attribute(N, T, V, H, s1)$ yields the result:

$$attribute(pf, column, 3, [output1, input1, pf\_1], s1)$$
$$attribute(pf, column, 2, [output2, input1, pf\_1], s1)$$

This example is simple, but real-life rules and workflows are lengthier, so people lose track of them for MIMO multi-staged data-flow graphs. Because situation calculus does not support parallel actions, topological sort is used to linearize the action sequence of graph-wide reasoning.

Our implementation performs retrospective analysis, using provenance as the *lingua franca* of data-flow graphs. It supports two distinct provenance schemes: CWLProv[1] and S-Prov[2]: CWLProv is developed by the CWL community, which is a file-oriented workflow system; S-Prov is a scheme for dispel4py, a fine-grained data-streaming workflow system. In order to support this we use an intermediate abstract representation of the data-flow graphs, and extract and convert the original provenance to that using SPARQL queries.

Our evaluation is based on two real-life scientific workflows: cyclone tracking and Moment Tensor in 3D (MT3D). We use the provenance generated by the execution of the corresponding scientific workflows, where some provide CWLProv and others S-Prov. The actual data-governance rules involved in the executions are encoded using our formal language to test our language's coverage; our framework's correctness is tested by performing reasoning over data-flow graphs extracted from provenance to obtain the activated obligations and the derived data-governance rules associated with each run's data products. We also encode a series of real-life data-governance rules, whether used in the experiments or not, to demonstrate the generality of the model.

3 Future Direction

Further work will be focused on expanding our language beyond obligations (for instance to include *prohibitions* checked *before* using the data), establishing a more formal link with other work (e.g. decentralized Information Flow Control) in order to connect with a wealth of existing work that uses it, and performing static optimization of flow rules to reduce the reasoning complexity and improve efficiency.

References

[Elnikety *et al.*, 2016] Eslam Elnikety, Aastha Mehta, Anjo Vahldiek-Oberwagner, Deepak Garg, and Peter Druschel. Thoth: Comprehensive Policy Compliance in Data Retrieval Systems. In *25th USENIX Security Symposium*, pages 637–654, 2016.

[Pasquier *et al.*, 2017] Thomas F. J.-M. Pasquier, Jatinder Singh, David Eyers, and Jean Bacon. CamFlow: Managed Data-sharing for Cloud Services. *IEEE Transactions on Cloud Computing*, 5(3):472–484, 2017.

[Zhao and Atkinson, 2019] Rui Zhao and Malcolm Atkinson. Towards a Computer-Interpretable Actionable Formal Model to Encode Data Governance Rules. In *15th International Conference on eScience*, pages 594–603, 2019.

[1] https://w3id.org/cwl/prov/

[2] https://github.com/aspinuso/s-provenance