A SHACL-based Data Consistency Solution for Contract Compliance Verification

Robert David[1,2], Albin Ahmeti[1,3], Geni Bushati[4], Amar Tauqeer[4,5] and Anna Fensel[5]

[1] *Semantic Web Company GmbH, Austria*
[2] *Vienna University of Economics and Business, Austria*
[3] *Vienna University of Technology, Austria*
[4] *Semantic Technology Institute, Department of Computer Science, Universität Innsbruck, Austria*
[5] *Artificial Intelligence Chair Group, Wageningen University & Research, The Netherlands*

Abstract

In recent years, there have been many developments for GDPR-compliant data access and sharing based on consent. For more complex data sharing scenarios, where consent might not be sufficient, many parties rely on contracts. Before a contract is signed, it must undergo the process of contract negotiation within the contract lifecycle, which consists of negotiating the obligations associated with the contract. Contract compliance verification (CCV) provides a means to verify whether a contract is GDPR-compliant, i.e., adheres to legal obligations and there are no violations. The rise of knowledge graph (KG) adoption, enabling semantic interoperability using well-defined semantics, allows CCV to be applied on KGs. In the scenario of different participants negotiating obligations, there is a need for data consistency to ensure that CCV is done correctly. Recent work introduced the automated contracting tool (ACT), a KG-based and ODRL-employing tool for GDPR CCV, which was developed in the Horizon 2020 project smashHit (https://smashhit.eu). In this work, we propose a novel approach to overcome some limitations of ACT. We semi-automatically resolve CCV inconsistencies by providing repair strategies, which automatically propose (optimal) solutions to the user to re-establish data consistency and thereby support them in managing GDPR-compliant contract lifecycle data. We have implemented the approach, integrated it into ACT and tested its correctness and performance against basic CCV consistency requirements.

Keywords

Privacy protection, GDPR, Contract compliance verification, Data consistency, Constraint languages, SHACL, Knowledge graphs, Logic programming, Answer set programming

1. Introduction

One of the legal bases that must be satisfied for General Data Protection Regulation (GDPR) [1] is *informed consent* before sharing any actual data. In certain cases, consent might not be sufficient and parties need to establish a *contract* for data sharing. This is typical for online services that comprise Business-to-Business (B2B) and Business-to-Consumer (B2C) data sharing, where a contract specifies the terms and obligations specifying each party's responsibilities. The contract lifecycle comprises all the phases, namely, negotiation, signing, execution, auditing, and termination/renewal phase. In this context, *Contract Compliance Verification (CCV)* is about auditing and provides a means to verify whether a contract is GDPR compliant, that is, to check and report violations, e.g., for not fulfilling an obligation.

Recently, approaches for digital contracting to tackle the CCV challenge are often based on knowledge graphs (KGs) [2] as a means to improve interoperability, interpretation, and contextualization of data, employing precise semantics through *ontologies* and *controlled vocabularies*. The automated contracting tool (ACT) [3] has been tested and used in two data sharing scenarios, namely, for car insurances and for smart cities. Especially in the latter case, a vast amount of data pertaining to contracts is shared, calling for a more scalable, trusted, and secure platform for data sharing. We define our challenges as:

ODRL and beyond: practical applications and challenges for policy-base access and usage control. Co-located with the Extended Semantic Web Conference, June 01 – 02, 2025, Portorož, Slovenia
robert.david@graphwise.ai (R. David); albin.ahmeti@graphwise.ai (A. Ahmeti); geni.bushati@sti2.at (G. Bushati); amar.tauqeer@wur.nl (A. Tauqeer); anna.fensel@wur.nl (A. Fensel)
0000-0002-3244-5341 (R. David); 0000-0001-8766-4069 (A. Ahmeti); 0000-0002-3345-387X (A. Tauqeer); 0000-0002-1391-7104 (A. Fensel)

(i) Identifying and reporting CCV violations by defining and integrating constraints for ACT using the Shapes Constraint Language SHACL.

(ii) Fixing CCV violations, i.e. semi-automatically repairing violations while at the same time ensuring the semantics of the contract.

To cope with these challenges, we first establish integrity constraints using SHACL. In the context of the ACT tool, it may occur that CCV violations are reported based on inconsistencies present in received data. For such cases of CCV violations, we employ SHACL validation as a declarative approach to identify inconsistencies, and *SHACL repairs* [4] to semi-automatically correct identified inconsistencies with a minimum of user intervention. This is achieved by defining and implementing repair strategies for inconsistent data reported by CCV. These repair strategies i) extend our previous work on SHACL repairs [4] to provide use case specific repair optimizations and ii) integrate with our ACT tool for users to semi-automatically fix inconsistent data reported by CCV.

2. Related Work

There are several systems developed to manage contracts consistently between participants, as described in [5] and [6]. Related work for automated contract management can be found in [7], [8] and [9]. Regulation compliance checking is shown in [10] using ontologies and SPARQL. Automated compliance checking for RDF data is presented in [11].

3. CCV Architecture

Contract Lifecycle The contracting process involves drafting, negotiating terms, signing, executing obligations, and performing CCV checks during the auditing phase. Finally, the process ends with the termination or renewal of the contract.

Data Model The data for contracts and related elements is described using the Contract Ontology (CO), which incorporates the Financial Industry Business Ontology (FIBO) to semantically describe contract lifecycle elements. This work specifically focuses on the *fibo:Contract* class, represented by the property *co:hasContractStatus*, and the *co:Obligation* class, where contracts are linked via the *co:hasObligations* property and the obligation state is represented by *co:hasState*.

ACT [3] is an application that is used to create and manage contracts between different parties along the contract lifecycle. Using its interface, users can associate clauses and obligations with contracts. The ACT UI flags whenever there is an obligation that has not been fulfilled in the respective due time by a party, i.e. GPDR compliance has been violated. When creating and changing a contract or obligation, respectively, CCV validation is done via SHACL processor. The validation process checks the conformance with respect to consistency requirements pertaining to obligations, such as if the status of a contract is consistent with the obligation states. Furthermore, ACT has been extended[1] to display whenever there are data inconsistencies, and it offers to restore consistency by selecting one of the repair choices via a human-in-the-loop approach.

4. CCV Consistency and Repairs

The CCV consistency requirements were originally determined for the contract data in the smashHit KG, which was built for the use cases of the smashHit project. For achieving CCV consistency in relation to

[1]https://github.com/GeniBushati/cmt

ACT, we introduce repair strategies to automatically ensure that every contract has a clearly defined and consistent status regarding the contract lifecycle.

Recent work presented SHACL repairs – an approach to repair RDF graphs to satisfy given SHACL constraints. It is based on explanations for constraint violations, which are represented as sets of additions and deletions for the RDF graph to achieve conformance. The implementation uses answer set programming (ASP) to provide repair choices as part of minimal models. In our context of fixing violations reported by CCV, we aim to automatically determine an optimal repair choice to minimize human-in-the-loop. In cases where this is not possible, the ACT UI has been extended to allow users to select one preferred choice.

Consequently, we extend SHACL repairs with repair strategies for ACT, which formalize optimizations to automatically determine optimal repair choices compliant to CCV consistency requirements. We implement a repair strategy program on top of the SHACL repair program, which parses RDF repair strategies and automatically generates additional ASP rules to determine optimal choices.

In the following, we provide an example for one of the ACT consistency requirements. The following SHACL shape represents the consistency requirement: "*A Contract is violated if at least one associated Obligation has the state set to violated*".

```
:ContractViolationShape a sh:NodeShape; sh:targetClass fibo:Contract;
  sh:property [ sh:path :hasContractStatus; sh:maxCount 1; ];
  sh:or (
        [ sh:not [ sh:property [ sh:path ( co:hasObligations co:hasState ); sh:hasValue co:ViolatedState; ]; ] ]
        [ sh:property [ sh:path co:hasContractStatus; sh:hasValue co:statusViolated; ] ] ).
```

Consistency for *ContractViolationShape* means that there must not be any non-violated contract if there is a violated obligation. In case of a violation, the repair program either deletes *hasObligations*, removes the *ViolatedState* from the obligations or it adds the *statusViolated* to the contract. The repair strategy we define for CCV semantics is to only allow adding *statusViolated* for *hasContractStatus* to represent the actual real-world situation. We define two constraints for the repair strategy.

```
:ViolatedContractStrategy a shr:RepairStrategy;
        shr:hasConstraint [ sh:path co:hasObligations; sh:action sh:delete; ];
        shr:hasConstraint [ sh:path co:hasState; shr:action shr:delete; shr:values ( co:ViolatedState; ) ] .
```

We assume the following data graph:

```
:contb2b a fibo:Contract; co:hasContractStatus :statusFulfilled; co:hasObligations :ob_1 .
:ob_1 a co:Obligation; co:hasState co:ViolatedState .
```

Without the repair strategy, the repair program would return the following two minimal repairs with deletions D_1 and D_2, which both violate the CCV semantics:

$$D_1 = \{hasObligations(contb2b, ob\_1)\} \qquad D_2 = \{hasState(ob\_1, ViolatedState)\}$$

In contrast, when we apply the repair strategy, the repair program will return a new minimal repair under the constraints added by the repair strategy with additions A and deletions D:

$$A = \{hasContractStatus(contb2b, statusViolated)\}$$
$$D = \{hasContractStatus(contb2b, statusFulfilled)\}$$

With repair strategies, the data can now only be repaired when assigning the contract status *statusViolated* to the contract, thereby implementing the expected CCV semantics.

Evaluation of CCV Repairs Tests were conducted using contract data from the smashHit project to evaluate the correctness and performance, including bulk performance tests. The results in Fig. 1 show that data with up to 3000 inconsistencies was repaired within a minute. The performance curve shows an above linear development. As a conclusion from the test results, our approach shows promising performance in practice, highlighting the approach's correctness and scalability. The implementation and test data are available on GitHub[2].

[2]https://github.com/robert-david/shacl-repairs/tree/ccv-repair-strategies

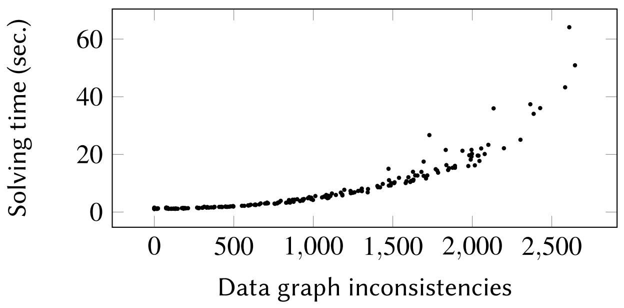

Figure 1: Bulk Performance Tests

5. Conclusions & Future Work

This work presents a new approach for semi-automatically resolving data inconsistencies in CCV using the ACT tool, simplifying the repair process and helping to manage GDPR-compliant contract data. Future work will focus on handling more complex constraints, evaluating repair quality, and expanding the approach to broader use cases, ultimately improving the applicability and reliability of semantic contract environments and supporting the adoption of formalized contracts in automated systems.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] B. Wolford, General Data Protection Regulation (GDPR)., Available online: https://gdpr.eu/what-is-gdpr/, 2022. Accessed on 20 July 2022.

[2] A. Tauqeer, A. Kurteva, T. R. Chhetri, A. Ahmeti, A. Fensel, Automated GDPR contract compliance verification using knowledge graphs, Inf. 13 (2022) 447. doi:10.3390/info13100447.

[3] A. Tauqeer, A. Fensel, GDPR data sharing contract management and compliance verification tool, Software Impacts 21 (2024) 100653. doi:doi.org/10.1016/j.simpa.2024.100653.

[4] S. Ahmetaj, R. David, A. Polleres, M. Šimkus, Repairing shacl constraint violations using answer set programming, in: The Semantic Web – ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2022, p. 375–391. doi:10.1007/978-3-031-19433-7_22.

[5] R. G. Brown, The corda platform: An introduction, Retrieved 27 (2018) 2018. doi:DOI:10.13140/RG.2.2.30487.37284.

[6] L. Guo, Q. Liu, K. Shi, Y. Gao, J. Luo, J. Chen, A blockchain-driven electronic contract management system for commodity procurement in electronic power industry, IEEE Access 9 (2021) 9473–9480. doi:10.1109/ACCESS.2021.3049562.

[7] J. Jiang, H. Aldewereld, V. Dignum, Y.-H. Tan, Compliance checking of organizational interactions, ACM Trans. Manage. Inf. Syst. 5 (2015). doi:10.1145/2629630.

[8] A. Lomuscio, H. Qu, M. Solanki, Towards verifying compliance in agent-based web service compositions, volume 1, 2008, pp. 265–272. doi:10.1145/1402383.1402424.

[9] C. Molina-Jimenez, S. Shrivastava, M. Strano, A model for checking contractual compliance of business interactions, IEEE Transactions on Services Computing 5 (2011) 276–289. doi:10.1109/TSC.2011.37.

[10] K. R. Bouzidi, B. Fies, C. Faron-Zucker, A. Zarli, N. L. Thanh, Semantic web approach to ease regulation compliance checking in construction industry, Future Internet 4 (2012) 830–851. doi:https://doi.org/10.3390/fi4030830.

[11] L. Robaldo, F. Pacenza, J. Zangari, R. Calegari, F. Calimeri, G. Siragusa, Efficient compliance checking of RDF data, Journal of Logic and Computation (2023). doi:10.1093/logcom/exad034.