Towards Contextualized Rule Repositories for the Semantic Web

Felix Burgstaller, Christoph G. Schuetz, Bernd Neumayr, Dieter Steiner, Michael Schrefl

Department for Business Informatics - Data & Knowledge Engineering
Johannes Kepler University Linz, Austria
firstname.surname@jku.at

Abstract. Central to the semantic web are ontologies: shared conceptualizations of domains of interest expressed in an ontology language such as OWL. Rule languages complement ontology languages. For large heterogeneous bodies of knowledge on the semantic web, contextualized knowledge repositories facilitate the organization of ontological concepts. In this paper we propose a similar mechanism – contextualized rule repositories – for organizing and executing large sets of context-specific rules. We investigate feasibility of a SPARQL Inferencing Notation (SPIN) based implementation using a real-world use case from the aeronautical domain. Furthermore, we compare SPIN-based contextualized rule repositories to a context-unaware SPIN implementation.

Keywords: Context, ontologies, business rules, SPIN

1 Introduction

Ontologies and ontology languages are at the heart of the semantic web. Ontologies are shared conceptualizations of domains of interest expressed in machine-readable formats, i.e., ontology languages [16]. Among the most common ontology languages for the semantic web are RDF(S) and OWL. Companies increasingly employ semantic web technologies to build semantic information systems, leading to corporate semantic webs [12].

Large heterogeneous knowledge repositories are rendered manageable by the introduction of *context*. A context constrains the validity of contextualized knowledge. Typically, contexts are hierarchically organized. A theoretical foundation for representing contextualized knowledge and reasoning about contexts was introduced by McCarthy [11]. The CYC project, a knowledge base containing 100,000 general concepts, employs contexts to organize its knowledge [10]. Contexts are also employed in the semantic web, e.g., the metaview approach [17] which introduces a framework to manage metalevel information of OWL axioms.

A more recent approach are contextualized knowledge repositories [15] where the meaning of ontological concepts depends on the context. Concepts propagate from more general contexts to more specific contexts. A context is characterized

by attributes from different context dimensions. For example, aeronautical concepts may be interpreted differently depending on geographic region and governing body. In this case *geographic region* and *governing body* are the dimensions that determine context.

In the semantic web stack [1], rules and rule languages complement ontologies and common ontology languages such as OWL, which typically trade expressiveness for general decidability. Rules enable, for instance, inference of properties or extra-logical operations such as calculations. In the corporate semantic web, rule languages may express business rules. A W3C recommendation for a rule language is the Rule Interchange Format (RIF) [18], an XML language primarily designed to exchange rules. RIF provides different dialects, catering to different needs in rule definition, e.g., production rules. The most common rule languages for the semantic web are SWRL and SPARQL-based languages.

Complementing ontologies organized in contextualized knowledge repositories by rules, it is sensible to organize the rules in a similar fashion to promote efficient management. Contextualized knowledge repositories, however, are not designed for organizing and executing rules. Thus, a mechanism for contextualization, tailored specifically to the particularities of rules, is necessary. Such a mechanism are contextualized rule repositories [5, 4].

In this paper, we investigate the potential of *contextualized rule repositories* as tool to organize and execute large sets of context-specific rules in the semantic web. We provide an implementation of contextualized business rules [5] using the semantic web technologies OWL and SPARQL Inferencing Notation (SPIN) [9]. Furthermore, we investigate, using a real-world use case from the aeronautical domain, the feasibility of a SPIN-based contextualized organization and execution of rules and compare it to a solution without contexts.

The remainder of this paper is structured as follows. Sect. 2 provides background on contextualized business rules and related work. Sect. 3 presents the demo case from the aeronautical domain used to illustrate the approach. Sect. 4 describes the implementation of contextualized rule repositories using SPIN. Sect. 5 presents results from feasibility experiments. We conclude the paper with a summary and an outlook on future work.

2 Background

In this section we present the conceptual framework for contextualized rule repositories [5] and related work.

2.1 Contextualized Business Rules

We previously introduced a generic conceptual context model for contextualized business rules [5]. This context model is multi-level, i.e., it can be instantiated to a specific domain, e.g., the aeronautical domain, and subsequently to a particular application within the domain, e.g., filtering aeronautical messages. A simplified version of this model is depicted in Fig. 1. The preceding "/" indicates a derived

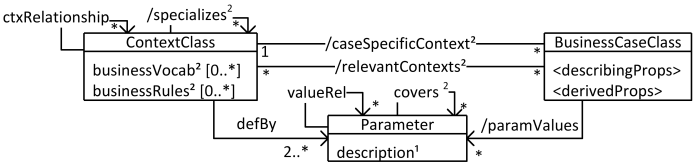

Fig. 1. The generic context model for business rule management.

relationship, the superscript numbers indicate the potency (default ^{1}), i.e., how many levels below the attribute or relationship must be instantiated.

`ContextClass` and `Parameter` are instantiated to domain-specific context classes and parameters on the domain level; on the application level they are instantiated to specific contexts and hierarchically ordered parameter values (`covers`). Specific contexts state the business rules (`businessRules`) and business vocabulary (`businessVocab`) applying for them. Parameter value hierarchies are used to derive the hierarchy of contexts (`specializes`). Business rules and vocabulary propagate along this `specializes` relationship. Besides these hierarchical relationships, it is possible to define further context and parameter value relationships by instantiating `ctxRelationship` or `valueRel` respectively.

For a specific business case (an instance of an instance of `BusinessCaseClass`) described by `<describingProps>`, we infer its parameter values (`paramValues`) for each `Parameter`. These parameter values identify the most specific context(s) relevant to the given business case. Using the hierarchy of contexts all relevant contexts can be determined (`relevantContexts`). Depending on the semantics of the employed context relationships, e.g., `specializes` inherits business rules and business vocabulary, conflicts, e.g., due to multi-inheritance, need to be resolved, e.g., more specific business rules/terms prevail. Therefore, a new context containing only relevant business rules and business vocabulary with conflicts resolved is derived (`caseSpecificContext`). The effects of business rules are modeled using the placeholder `<derivedProps>`.

2.2 Related Work

A Contextualized Knowledge Repository (CKR [15]) for the semantic web organizes ontological concepts in a space of hierarchically-ordered contexts. Each context is characterized by a set of attributes from different context dimensions. The attributes of the context dimensions can be hierarchically ordered. The hierarchical order of contexts is derived from the hierarchical order of the dimension attributes. Concepts of more general contexts propagate to the more specific contexts. The same concept may assume a different meaning in different contexts. Since rules in the semantic web are commonly represented using OWL or RDF(S), a CKR may also serve to organize sets of rules. In that case, however, rule execution must be taken care of by another component. Furthermore, CKRs lack a mechanism for automatic determination of relevant contexts for a business situation that initiates rule execution.

Apart from the semantic web, many commercial business rule management systems (BRMSs) offer the possibility to organize rules into rulesets. These commercial systems typically lack a mechanism for declaratively describing which ruleset should be invoked in which situation. Concerning the representation of relationships between rulesets, only inclusion is commonly supported.

Other approaches propose grouping of business rules by single attributes, e.g., regarding the rule objective [7] or regarding business dimensions like concerned business objects [13]. More sophisticated approaches organize rules by multiple attributes, e.g., by situation [14] or linking business rules with goals [8]. These systems typically lack support for free definition of attributes, and often support only inclusion relationships between rulesets. Thus, compared to these approaches contextualized rule repositories are more flexible with respect to context parameters and enable automatic context-specific execution of rules.

3 Use Case: Classification of Aeronautical Messages

We previously demonstrated usefulness of contextualized business rules in the aeronautical domain [5] as part of the Semantic NOTAM research project (SemNOTAM [3]). In particular, we applied contextualized rules to relevance and importance classification of aeronautical messages for flight operations personnel. SemNOTAM must be seen in the wider context of System Wide Information Management (SWIM [6]). SWIM's vision is to create a shared understanding of information and to provide relevant information to SWIM consumers. In this sense, SWIM can be considered a corporate semantic web application for the aeronautical domain. Relevance and importance classification of aeronautical messages is essentially classification by rules which can be represented using semantic web rule languages. We employ SemNOTAM as use case for a feasibility study of contextualized rule repositories in the semantic web.

Apart from the aeronautical domain, the proposed approach can be applied to other domains as well. For example, a person's eligibility for retirement depends on the country of residence, country of workplace, age, years of work, etc. Furthermore, the computation of monthly retirement payment differs.

Fig. 2 (upper part) illustrates a simple domain-specific model for SemNOTAM instantiating the general model in Fig. 1. This model focuses on classification of highly important NOTAMs only (`highImportance`). Aeronautical Information Management (AIM) contexts (`AIMContext`) are, in our use case, described by two parameters of seven identified so far in pilot interviews: meteorological condition (`MeteoCond`) and aircraft type (`AircraftType`). Other parameters identified include the role of the user and the aviation type (e.g. cargo, passenger, military, etc.). In SemNOTAM, only inheritance relationships are of importance, i.e., `covers` and `specializes`. The `BusinessCaseClass` interest specification (`InterestSpec`) describes a flight plan, with the describing properties weather (`weather`) and type of aircraft used (`aircraft`), for which messages have to be grouped by importance. Note that temporal and spatial attributes, e.g., the used flight segments, are not shown for comprehensibility.

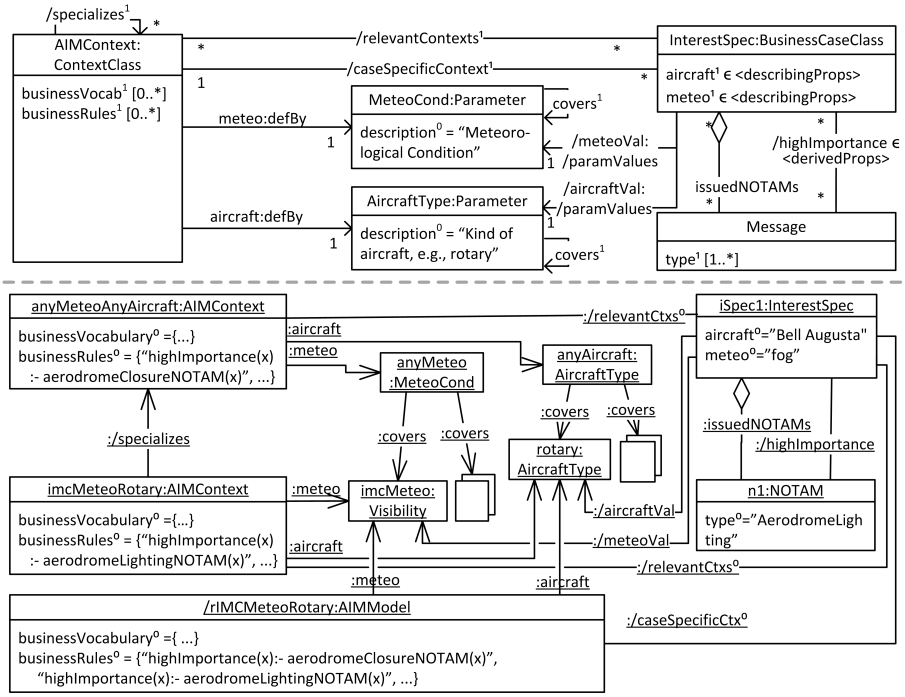

Fig. 2. A context model for rule-based message filtering in the aeronautical domain and a concrete application instantiation.

We instantiate the presented domain-specific context model to an application-specific model in Fig. 2 (lower part). In particular, we define two exemplary contexts: `anyMeteoAnyAircraft` and its specializing context `imcMeteoRotary`. Context `anyMeteoAnyAircraft` contains rules and vocabulary applying for any interest specification whereas rules and vocabulary in context `imcMeteoRotary` apply only for interest specifications regarding instrumental meteorological conditions (IMC) and rotary aircrafts. Besides these contexts, we specify a concrete interest specification `iSpec1` with aircraft Bell Augusta, a helicopter, and weather fog, a specific IMC. Using this information, the contexts relevant to `ispec1` can be derived: `anyMeteoAnyAircraft` and `imcMeteoRotary`. The case-specific context `rIMCMeteoRotary` is, as contexts `anyMeteoAnyAircraft` and `imcMeteoRotary` do not conflict, the union of these two contexts. Notam `n1` is determined to be of `highImportance` due to the rule in context `imcMeteoRotary`.

4 Contextualized Rule Repositories using SPIN

SPIN is a SPARQL-based rule and constraint language integrating concepts from rule-based systems, query languages, and object-oriented languages. SPIN enables formal description of object behavior, i.e., SPIN statements are assigned

```
CONSTRUCT { spin:_this highImportance ?n. }
WHERE { ?n rdf:type notams:AerodromeLightingNOTAM. }
```

Listing 1. The rule from context `imcMeteoRotary`.

```
CONSTRUCT { spin:_this highImportance ?n. }
WHERE { spin:_this meteo some MeteoCond.
        spin:_this aircraft some RotaryAircraft.
        ?n rdf:type notams:AerodromeLightingNOTAM. }
```

Listing 2. The rule from context `imcMeteoRotary` in its context-unaware form.

to classes and apply to individuals of their assigned classes only. The SPIN W3C member submission [9] comprises the SPIN modeling vocabulary and an RDF representation of SPARQL. The SPIN modeling vocabulary supports inference rules (`spin:rule`), constraints on classes (`spin:constraints`), and constructors (`spin:constructor`). These are specified using SPIN's RDF representation of SPARQL. For instance, the SPARQL representation of the rule from our use case defining aerodrome closure NOTAMs (`AerodromeLightingNOTAM`) to be of high importance (`highImportance`) within context `IMCMeteoRotary` is shown in Listing 1; Listing 2 depicts the corresponding context-unaware rule. SPIN rules can be organized into rule libraries. Furthermore, the SPIN modeling vocabulary supports user-defined execution orders of SPIN-rules, user-defined rule templates, and user-defined functions. The latter two features increase rule readability and promote reuse of rules. Implementing contextualized rule repositories using SWRL would require workarounds as for example rules cannot be assigned to classes and thus rule execution of only relevant rules is difficult. Furthermore, since we consider businesses, the use of standards, such as SPARQL, is preferable.

A common implementation of SPIN is TopBraid's Jena-based SPIN API[1]. In order to evaluate SPIN-rules, the corresponding RDFS/OWL ontology needs first to be loaded and subsequently prepared for rule inference. Any changes to SPIN rules in an ontology require a new inference preparation. Any other changes, e.g., new individuals, even if resulting from SPIN rules, are immediately considered in the current SPIN rule evaluation.

In order to implement contextualized rules using SPIN, context classes and contexts, their parameters and parameter values, as well as business case classes and business cases need to be represented in RDF(S) or OWL. Each context, utilizing SPIN's object orientation, is represented as an OWL class with its rules assigned, e.g., Listing 3 depicts context `IMCMeteoRotary` from our use case. Parameters and their parameter values form subsumption hierarchies. The assignment of parameter values to contexts can be represented in two ways. Regardless the variant, the hierarchy of contexts can be derived from the parameter value hierarchies using a reasoner. The two ontology variants are:

1. *Ctx*: employing object properties (Listing 3) as modeled in [5, 4] and
2. *SCtx*: subclassing (Listing 4) presumably faster in evaluation.

[1] `http://topbraid.org/spin/api/`

```
Class: IMCMeteoRotary
  EquivalentTo:
    InterestSpec
      and meteo some MeteoCond
      and aircraft some RotaryAircraft
  SubClassOf:
    Context
```

Listing 3. Context `IMCMeteoRotary` with object properties (Ctx).

```
Class: IMCMeteoRotary
  EquivalentTo:
    InterestSpec
      and IMCMeteo
      and RotaryAircraft
  SubClassOf:
    Context

Individual: IMCMeteoRotary
  Facts:
    spin:rule _:1
```

Listing 4. Context `IMCMeteoRotary` with subclassing (SCtx) including a SPIN rule.

Each business case class is represented as an OWL class and defines its describing properties as contexts do, i.e., either by subclassing or using object properties. In the simplest configuration, describing properties instantiate or refer to the parameter values directly. In more complex configurations, parameter values must first be derived from describing properties using OWL or SPIN.

This approach enables an OWL reasoner to determine the relevant contexts for a given business case. For instance, interest specification `iSpec1` (Ctx) relates to individuals `BellAugusta` (instance of `Rotary`) and `Fog` (instance of `IMCMeteo`) using the object properties `aircraft` and `meteo` respectively. Thus, `iSpec1` is reasoned to be an individual of `IMCMeteoRotary` by Listing 3. As `IMCMeteoRotary` is a subclass of `AnyMeteoAnyAircraft` by hierarchies of parameter values, `iSpec1` is an individual of `AnyMeteoAnyAircraft` as well. Since a business case can be an individual of multiple contexts, the rules of several contexts are evaluated when performing SPIN evaluation. Thus, conflicting conclusions can be derived, e.g., a NOTAM could be classified both relevant and irrelevant. To resolve this, a cautious resolution rule could be defined: a NOTAM is relevant as soon as one rule classifies it as relevant. In other cases, most specific rules may have precedence. A detailed analysis of conflicts and their resolutions is considered future work.

Contextualized business rules [5] also consider business vocabulary. Thus, SPIN-based contextualized rule repositories should enable contextualized vocabulary as well. One option is to represent the entire vocabulary using SPIN rules. In this case, however, a reasoner would have to first execute the vocabulary rules before making inferences. Modeling vocabulary explicitly, one ontology per context containing the context's vocabulary is necessary. These ontologies must be imported in Ctx and SCtx respectively to enable their use in SPIN rules, i.e.,

```
Individual: iSpec1
  Types:
    InterestSpec
  Facts:
    aircraft BellAugusta,
    meteo Fog
```

Listing 5. The Manchester representation of interest specification `iSpec1` (Ctx).

all vocabularies are accessible in all contexts. Creating one SPIN rule library per context would theoretically allow to define terms local to a context by defining them in the corresponding rule library. In practice, however, non-SPIN triples relevant to rule execution are not allowed in SPIN rule libraries. Another option, left for future work, is to integrate contextualized knowledge repositories [15].

We expect the proposed SPIN implementation of contextualized rule repositories to outperform context-unaware solutions. Nevertheless, as this implementation introduces additional organizational overhead, we suspect that this is only the case if the number of rules and contexts exceeds some threshold. Thus, we postulate the following hypotheses regarding SPIN-based contextualized rule repositories which we evaluate in the following section.

H1 The system applies for given business cases all and only relevant rules.
H2 There exists a threshold above which the average response time (the time between providing a business case and being returned the business case with rules applied) is lower compared to SPIN-based context-unaware solutions.

5 Feasibility Study

To demonstrate the feasibility of our proposed SPIN-based contextualized rule repositories for the semantic web, we evaluate our hypotheses utilizing SemNOTAM as use case. To this end, we employ TopBraid's single-threaded SPIN API. To test hypothesis H1, we consider simple rules of the kind "In <situation/context>, messages of <type> are of <level> importance" as well as more complex rules which include, for instance, restrictions on aircraft characteristics like its weight or wingspan. To test hypothesis H2 we employ simple rules only as we assume that the complexity of rules influences context-aware and context-unaware ontology variants alike; an empirical evaluation of this assumption is outstanding. Regarding interest specifications we assume the simple configuration, i.e., parameter values are directly assigned. The complex configuration requires additional rules which derive the parameter values. Using SPIN's ability to define an execution order of rules, the only difference in the configurations is the total number of rules affecting all ontology variants to the same extent.

Hypothesis H1 To test H1, we represented our use case using SPIN-based contextualized rule repositories. We employed four parameters, namely, flight phase, flight rule, meteorological condition, and time of day, of the seven we have

identified in pilot interviews. Each of these parameters defines three parameter values, i.e., one root element and two children, resulting in $3 \times 3 \times 3 \times 3 = 81$ potential contexts for which we can define specific rules. So far, we have elicited rules for 17 of these 81 contexts regarding 22 different aeronautical message types. Testing various interest specifications covering the 81 potential contexts, we could verify that *only relevant* as well as *all relevant* rules were applied. Thus, hypothesis H1 is supported by our experiments.

Hypothesis H2 To test hypothesis H2 we derive context-unaware ontology variants from our contextualized ontology variants SCtx and Ctx and represent them using SPIN. Therefore, we incorporate the parameter values of each context into its contained rules, i.e., the rule in Listing 1 is rewritten to the rule in Listing 2. Consequently, for a single context model we have two context-aware variants (*SCtx* and *Ctx*) and two context-unaware variants (*SNoCtx* and *NoCtx*).

A single context model is not sufficient to test hypothesis H2, a generator is needed to create context models of various sizes and complexity and their corresponding ontology variants. Furthermore, random interest specifications need to be generated in order to find the suspected threshold. Consequently, we constructed a generator accepting six parameters determining the size and complexity of the generated ontologies: (1) number of parameters, i.e., instances of `Parameter` in Fig. 1 (two in our use case in Fig. 2), (2) parameter value hierarchy depth, i.e., number of levels of `covers` (two plus in Fig. 2), (3) parameter value hierarchy width, i.e., instances of `covers` at one level (two plus in Fig. 2), (4) context density, i.e., the percentage of potential contexts actually instantiated (2 in 4 for the named parameter values and contexts in Fig. 2), (5) number of rules per context, i.e., cardinality of attribute `businessRules` (one plus in Fig. 2), (6) and number of message types (one in Fig. 2).

For the tests we used Intel Core i7-4770s with 16 GB RAM. We ran the generator for different configurations covering number of parameters (2-5), parameter value hierarchy depth (1-2) and width (2-5), context density (25 %, 50 %, 75 %), rules per context (12, 25, 50, 100), and number of message types (70). The latter parameter was not varied as it had no direct influence on the number of rules or contexts. The configurations were chosen in such a way that we tested up to about 12,000 contexts. Tests of Ctx were excluded due to its weak performance in preliminary tests. Ontology metrics for two sample context models are depicted in Table 1 . We measured the time for loading and for SPIN inference preparation as well as the average response time for 25 randomly generated interest specifications for the ontology variants SCtx, SNoCtx, and NoCtx.

Regarding loading time and inference preparation times we expect SCtx to perform best as its ontology variants are the smallest and its rules are the simplest, i.e., they contain the fewest terms. Analyzing the loading times for the different ontology variants in detail showed that the average time mainly depends on the total number of rules ($totalRules$). The time (in seconds) necessary for loading an SCtx ontology variant can be predicted using $avgTime = 2.15 \times 10^{-4} \times totalRules$. This linear regression has an R^2 value of 0.99, i.e., 99 % of the

Table 1. Ontology variant metrics for two sample context models.

Ontology Variant	# Contexts	# Rules	# OWL Axioms	Size
SCtx (S)NoCtx	16	160	3,300 (4,200) 6,000	53.7 kB (78.7 kB) 114.7 kB
SCtx (S)NoCtx	1,800	180,000	2,200,000 (4,300,000) 7,900,000	53 MB (111 MB) 187 MB

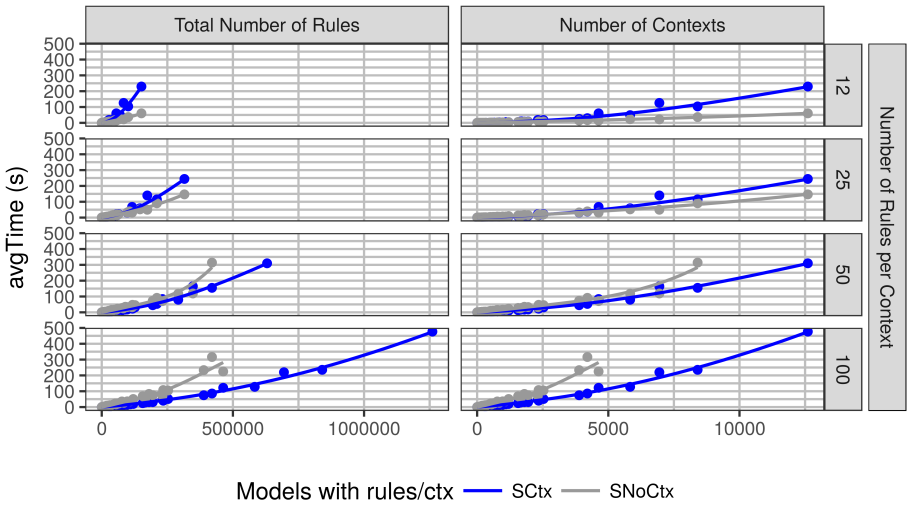

Fig. 3. Average response time vs. total number of rules grouped by rules per context.

variance in loading time is explained by it. SNoCtx takes about twice as long and NoCtx about 3.5 times as long. The time needed for inference preparation also mainly depends on the number of rules: SCtx $avgTime = 1.6 \times 10^{-4} \times totalRules$ ($R^2 = .99$), SNoCtx about thrice as long and NoCtx about 3.75 times as long. Consequently, the SCtx ontology variant is fastest considering loading and inference preparation confirming our expectation.

Considering the average response time, we expect SCtx to outperform NoCtx and SNoCtx for larger number of rules and contexts. Fig. 3 depicts the average response time versus the total number of rules (left column) and versus the number of contexts (right column) grouped by the number of rules per context. Note that due to the additional expressions in rules for context-unaware ontologies, tests of these ran out of memory when using more than 450,000 rules. Furthermore, NoCtx, surpassing SCtx only for 12 rules per context and performing worse than SNoCtx in any case, is not shown for readability. The left column in Fig. 3 reveals that the average response time of context-unaware ontologies only slightly depends on the number of rules per context. For SCtx, on the other

hand, there is a strong dependency. The threshold above which SCtx's average response time surpasses SNoCtx's is about 40 rules per context. For rules-per-context ratios below 40, the additional administrative effort seems to outweigh the benefits. Depicting the average response time versus the number of contexts (right column) we see that SCtx and SNoCtx for 25 rules per context have almost identical average response times up to approximately 3,000 contexts. Furthermore, we can see that the more contexts are employed the larger the gap between the average response time of SCtx and SNoCtx becomes.

In conclusion, SCtx outperforms SNoCtx and NoCtx in any case regarding loading time and inference preparation time. Regarding the average response time, SCtx outperforms SNoCtx when more than 40 rules per context are used. Thus, hypothesis H2 is supported by our experiments.

Predicting the Average Response Time Employing Pearson's correlation coefficient we found the average time for SCtx significantly correlated with the total number of rules ($\rho = .91$) and the number of used contexts ($\rho = .92$) most. Performing linear regression using these variables we are able to predict the average time for SCtx (in seconds) with $R^2 = .97$ (Equation 1).

$$\begin{aligned} avgTime_{\text{SCtx}} = -1.64 + 1.57 \times 10^{-5} \times totalRules + \\ 1.14 \times 10^{2} \times numCtx + 2.05 \times 10^{-8} \times totalRules \times numCtx \end{aligned} \tag{1}$$

6 Conclusion

We implemented contextualized rule repositories for the semantic web using SPIN, based on previous work on contextualized business rules [5]. This SPIN implementation feeds inferred knowledge directly into the ontology. Knowledge inferred by rules is then available for ontological and rule based reasoning.

Contextualized rule repositories using SPIN render large rulesets manageable. Concerning execution performance the granularity of the contexts matters. Fine granularity of contexts negatively affects, coarse granularity of contexts positively affects average runtime of rulesets. In our use case the threshold for positive impact of context granularity was approximately 40 rules per context. The threshold may vary depending on the use case, rule complexity, hardware configuration, and software configuration.

The current SPIN implementation of the contextualized rule repository realizes only a subset of the contextualized business rule model. Future work will investigate the implementation of additional relationships for contexts and parameter values as well as modification operations [4]. Worth further consideration is the integration of contextualized rule repositories in the formal framework of contextualized knowledge repositories [15]. Furthermore, we consider implementations using different rule languages such as SWRL and SPRINGLES [2].

Acknowledgments. Supported by the Austrian Ministry of Transport, Innovation, and Technology under grant FFG-839006 (SemNOTAM).

References

1. Berners-Lee, T.: Semantic web and linked data (2009), https://www.w3.org/2009/Talks/0120-campus-party-tbl/
2. Bozzato, L., Serafini, L.: Materialization calculus for contexts in the semantic web. In: Informal Proceedings of the 26th International Workshop on Description Logics. pp. 552–572 (2013)
3. Burgstaller, F., Steiner, D., Schrefl, M., Gringinger, E., Wilson, S., van der Stricht, S.: AIRM-based, Fine-grained Semantic Filtering of Notices to Airmen. In: Integrated Communication, Navigation, and Surveillance Conference (ICNS). pp. D3–1–D3–13 (2015)
4. Burgstaller, F., Neumayr, B., Schuetz, C.G., Schrefl, M.: Modification operations for context-aware business rule management. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC) (2017), (in press)
5. Burgstaller, F., Steiner, D., Schrefl, M.: Modeling Context for Business Rule Management. In: 2016 IEEE 18th Conference on Business Informatics (CBI). pp. 262–271 (2016)
6. Eurocontrol: System Wide Information Management (SWIM) (2017), http://www.eurocontrol.int/swim
7. Hamza, H., Fayad, M.: A novel approach for managing and reusing business rules in business architectures. In: The 3rd ACS/IEEE International Conference onComputer Systems and Applications. pp. 973–978 (2005)
8. Kardasis, P., Loucopoulos, P.: Expressing and organising business rules. Information and Software Technology 46(11), 701–718 (2004)
9. Knublauch, H., Hendler, J.A., Idehen, K.: SPIN - Overview and Motivation (2011), https://www.w3.org/Submission/spin-overview/
10. Lenat, D.B.: CYC: A large-scale investment in knowledge infrastructure. Commun. ACM 38(11), 33–38 (1995)
11. McCarthy, J.: Notes on formalizing context. In: Proceedings of the 13th International Joint Conference on Artifical Intelligence. vol. 1, pp. 555–560 (1993)
12. Paschke, A., Coskun, G., Heese, R., Luczak-Rösch, M., Oldakowski, R., Schäfermeier, R., Streibel, O.: Corporate Semantic Web: Towards the Deployment of Semantic Technologies in Enterprises, pp. 105–131. Springer, Boston (2010)
13. Rai, V.K., Anantaram, C.: Structuring business rules interactions. Electronic Commerce Research and Applications 3(1), 54–73 (2004)
14. Schäfer, A., Kreher, M.: Wie strukturiere ich meine Business Rules für eine effektive Pflege und Steuerung von kritischen, komplexen und dynamischen Geschäftsprozessen. In: GI Jahrestagung, vol. 1, pp. 213–218 (2010)
15. Serafini, L., Homola, M.: Contextualized knowledge repositories for the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 12–13, 64–87 (2012)
16. Staab, S., Studer, R.: Handbook on Ontologies. Springer Publishing Company, Incorporated, 2nd edn. (2009)
17. Tran, T., Haase, P., Motik, B., Grau, B.C., Horrocks, I.: Metalevel information in ontology-based applications. In: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2. pp. 1237–1242 (2008)
18. World Wide Web Consortium (W3C): RIF Overview (Second Edition) (2013), https://www.w3.org/TR/rif-overview/