Detection of Inconsistencies in Rules due to Changes in Ontologies: Let's Get Formal

Bruno Berstel-Da Silva[1] and Amina Chniti[2]

[1] Institut für Informatik
Albert-Ludwigs-Universität Freiburg
Germany
[2] INSERM UMRS 872 Eq.20,
Ingénierie des Connaissances en Santé,
Paris, France

Abstract In this paper, we focus on the impact of ontology changes on production rules, in the context of rule programs written over the entities of OWL ontologies. Then, ontology evolutions may make rules inconsistent with respect to the knowledge modeled by the ontology. To address this problem, we propose to combine two approaches: the syntactic approach of the $\mathcal{M}$odel-$\mathcal{D}$etect-$\mathcal{R}$epair ($\mathcal{MDR}$) method, and a semantic approach based on a formal description of production rules and rule program inconsistencies. The present paper shows on simple use cases the expected benefits of such a combination, which relies on existing implementations.
Keywords: Ontology, Production Rule, Ontology Change, Inconsistency, Program Analysis, Program Verification

1 Introduction

The integration of ontologies and rules is a topic of great interest for the semantic web community [1]. Interest has also been demonstrated by companies that care about ensuring the flexibility of their information systems [11]. In this context, a solution for authoring and executing production rules over OWL ontologies is proposed in [5]. Such an approach leverages the best of both worlds: an ontology is used to model the domain of concern, while a production rules program implements the computations of the business application.

In this short paper, we focus on the impact that a change in the ontology may have on the consistency of the rule program with respect to the knowledge modeled by the ontology. Indeed a change in the structure or in the TBox of the ontology can cause inconsistencies in the rule program, either at the rule level (self-contradiction, invariant violation...) or at the rule program level (subsumption, non-confluence...). To address these problems, we have developed in previous works [6] a new method, called $\mathcal{M}$odel-$\mathcal{D}$etect-$\mathcal{R}$epair ($\mathcal{MDR}$), to maintain the consistency of rules when an ontology evolves. The general idea of $\mathcal{MDR}$ consists in modeling the possible ontology changes, detecting the inconsistencies that can be caused by such changes, and proposing solutions, or “repairs”, to resolve the inconsistencies.

However, the “Detect” part in this method is based on the syntactic analysis of the rules. This already gives good results in simple situations, but suffers from the lack of a formal definition of inconsistencies. As a consequence, the accuracy of detection rapidly decreases with the complexity of the rules, and/or of the inconsistencies being looked for.

The present paper proposes to complement the $\mathcal{MDR}$ method with the formalization of inconsistencies in production rule programs that we describe in [2, 3, 4]. We expose this proposal the following sections, and we illustrate it on simple use cases, where a change in the domain of a property in the ontology causes the violation of an invariant by a rule.

2 The Problem

As mentioned above, we are interested in the problem of detecting inconsistencies in rules that are defined over ontologies, when these ontologies change. Our use cases are taken from a fictional market segmentation application, in which customers are assigned scores and categories.

Rule r_1 below assigns a `Bronze` category to a person p, when their score is lower than 20.

$$\mathtt{r_1(p:Person)} : \mathtt{p.score} < 20 \rightarrow \mathtt{p.category} \coloneqq \mathtt{Bronze}$$

Formally, a rule is a tuple $r = (\vec{o}, \vec{T}, g, a)$, where $\vec{o}$ are the variables for the objects matched by the rule, $\vec{T}$ are their types, and g and a are the condition[1] and the action of the rule. A rule condition is a non-quantified first-order logic formula in the rule variables; a rule action is a loop-free imperative program using the same variables.

In the example of rule r_1, we have $\vec{o} = (p)$, $\vec{T} = (\text{Person})$, $g \equiv p.score < 20$, and $a \equiv p.category \coloneqq \mathtt{Bronze}$. The rule r_1 is authored over an OWL ontology [5]. This ontology defines the concept Person that has the property *score* and the property *category* with an enumerated domain `category owl:oneOf {Bronze,Silver,Gold,Platinum}`. The rule r_1 includes an assignment that complies with the domain.

A change occurs in the ontology, by which the value `Bronze` is removed from the domain. This ontology change will make the rule r_1 inconsistent, as the value assigned to property *category* no longer is in the domain of the property.

Thus, we propose the $\mathcal{MDR}$ approach that enables to track ontology changes and to detect their impacts on rules authored over the ontology.

3 The $\mathcal{MDR}$ Approach

The $\mathcal{MDR}$ approach (for $\mathcal{M}$odel-$\mathcal{D}$etect-$\mathcal{R}$epair) consists in modeling ontology changes, detecting rule inconsistencies caused by the changes, and repairing the

[1] The letter g stands for ‘guard’.

detected inconsistencies [6]. This approach is based on Change Management Patterns (CMP) [8]. It consists of three categories of patterns: change patterns, inconsistency patterns and repair patterns. In this paper, we focus on the detection of inconsistencies, and hence on the inconsistency patterns.

Inconsistency detection in rules is performed by (1) detecting the rules impacted by an ontology change, and (2) detecting the rule inconsistencies caused by the change.

The impacted rules are detected by analyzing the syntax tree of each rule in the rule program in order to see if they use the ontology entity on which the change operates. For example, in the use case described in Section 2, the impacted rules are those using the property *category* in their definition.

The inconsistency patterns are implemented using *inconsistency detection rules* (IDRs). These rules express conditions for ontology changes to cause some kinds of inconsistency in rules. For example, an ontology change by which a value is removed from the enumerated domain of a property may cause inconsistencies of the kind *Rule Never Applies* and of the kind *Domain Violation*. The former can be detected when a rule includes an equality test with the value being removed in its condition. The latter can be detected when a rule assigns the value being removed in its action, as in the example rule of Section 2.

The inconsistency detection rules $\mathsf{RNA_{syn}}$ and $\mathsf{DV_{syn}}$ below implement these detection patterns:

```
$\mathsf{RNA_{syn}}(\mathsf{c : DomainChange, r : Rule, t : Comparison}) :$
    $\mathsf{r}$ in $\mathsf{c.impactedRules}$
        & $\mathsf{t}$ in $\mathsf{r.conditionTests}$
        & $\mathsf{t.operator = Equals}$
        & $\mathsf{t.property = c.property}$
        & $\mathsf{t.expression = c.valueRemoved}$
    $\rightarrow$ signal *Rule Never Applies* by $\mathsf{r}$

$\mathsf{DV_{syn}}(\mathsf{c : DomainChange, r : Rule, a : Assignment}) :$
    $\mathsf{r}$ in $\mathsf{c.impactedRules}$
        & $\mathsf{a}$ in $\mathsf{r.actionStatements}$
        & $\mathsf{a.propertyInLHS = c.property}$
        & $\mathsf{a.expressionInRHS = c.valueRemoved}$
    $\rightarrow$ signal *Domain Violation* by $\mathsf{r}$
```

However, this approach is based on the syntactic analysis of the rules, and is only correct up to the power of this syntactic analysis. For example, rule r_3 below would falsely be detected by $\mathsf{RNA_{syn}}$ as causing *Rule Never Applies*.

$$\mathtt{r}_3(\mathsf{p : Person}) : \mathtt{p.category} = \mathtt{Bronze}\ \textsc{or}\ \mathtt{p.category} = \mathtt{Silver} \rightarrow \mathtt{p.bonus} \coloneqq 5$$

As another illustration of the limits of a syntactic approach, $\mathsf{DV_{syn}}$ is limited to enumerated domains. Assume for example that the ontology defines the domain of the property *bonus* to $[0, 40]$ and the domain of *score* to $[0, 30]$. Rule r_2 below, which sets the score of a person p based on their bonus, complies with these domains. However, it no longer does if the domain of *bonus* is changed to $[0, 50]$ in the ontology.

$$\mathtt{r}_2(\mathtt{p}:\mathtt{Person}): \mathtt{p.score} = 0 \rightarrow \mathtt{p.score} \coloneqq (\mathtt{p.bonus} + 20) \div 2$$

It requires more than the syntactic analysis currently performed by $\mathcal{MDR}$ to detect such a *Domain Violation*.

4 A Formal Complement to the $\mathcal{MDR}$ Approach

In this section, we leverage the formal framework briefly exposed in Section 2, and in more details in [3, 4], to describe, and reason about, rules and rule programs. In this framework, the example rule r_2 is written $\vec{o} = (p)$, $\vec{T} = (\text{Person})$, $g \equiv p.score = 0$, and $a \equiv p.score \coloneqq (p.bonus + 20) \div 2$.

For the sake of simplicity, we assume in this paper that the action of a rule consists of a single assignment, that is, $a \equiv o.f_r \coloneqq e$ with $f_r \in \mathsf{Attr}$, where Attr is the set of the attribute symbols denoting the properties defined by the ontology, such as *category*, *score* or *bonus*. Because the formal framework defines rule semantics based on Hoare triples, this limitation can be lifted by using the results of Hoare logic.

When the ontology defines a domain on a property f of a class T, we represent this domain constraint with a closed formula $\Delta_f \equiv \forall o : T\ \delta_f$, where δ_f is a formula in the sole variable o, and involves only the attribute f. In our example, the formulas for the property domains after the changes in the ontology are

$$\begin{aligned} \Delta_{category} &\equiv \forall o : \text{Person}\,(o.category \in \{\mathtt{Silver}, \mathtt{Gold}, \mathtt{Platinum}\}) \\ \Delta_{score} &\equiv \forall o : \text{Person}\,(o.score \in [0, 30]) \\ \Delta_{bonus} &\equiv \forall o : \text{Person}\,(o.bonus \in [0, 50])\ . \end{aligned}$$

A rule r complies with these domains if, provided its condition is compatible with the domain formulas, the result of its execution still is compatible with them. In terms of Hoare triples, this can be expressed as $\{\bigwedge_{f \in \mathsf{Attr}} \Delta_f\}\, r\, \{\bigwedge_{f \in \mathsf{Attr}} \Delta_f\}$. That is, using the Hoare logic axiom for assignment:

$$\mathsf{CompliesWithDomains}(r) \overset{\text{def}}{\equiv} \bigwedge_{f \in \mathsf{Attr}} \Delta_f \wedge g \wedge \Delta_{f_r}[e/o.f_r]\ .$$

Conversely, a *Domain Violation* inconsistency occurs if, although the rule condition is compatible with the domain formulas, the result of its execution no longer is. This gives the following definition, illustrated further below on rule r_2:

$$\mathsf{DomainViolation}(r) \overset{\text{def}}{\equiv} \bigwedge_{f \in \mathsf{Attr}} \Delta_f \wedge g \wedge \neg\Delta_{f_r}[e/o.f_r]\ .$$

In this setting, the second inconsistency detection rule of Section 3 can be rewritten as follows to use $\mathsf{DomainViolation}$ as a semantic criterion:

```
$\mathsf{DV}_{\mathsf{sem}}(\mathsf{c : DomainChange, r : Rule}) :$
    $\mathsf{r}$ in $\mathsf{c.impactedRules}$ & $\mathsf{DomainViolation(r)}$
    $\rightarrow$ signal *Domain Violation* by $\mathsf{r}$
```

In this version of the inconsistency detection rule, we could omit the syntactic condition r `in` $\mathsf{c.impactedRules}$, as the semantic condition $\mathsf{DomainViolation(r)}$ is enough to detect the inconsistency. It can also be kept as an optimization.

For such a rule as $\mathsf{DV}_{\mathsf{sem}}$ to be executed, a representation of the rules such as r_1 and r_2 in our formal framework has to be built, and the inconsistency detection engine has to be able to assess the satisfiability of a formula such as $\mathsf{DomainViolation}$ [2]. This has been implemented in the Rule Static Analysis features of the ILOG BRMS[2] product, now known as IBM ODM[3] [10].

The inconsistency caused in r_2 by the extension of the domain of *bonus* is thus detected by assessing the satisfiability of $\mathsf{DomainViolation}(r_2)$, which reads:

$$\Delta_{category} \wedge \Delta_{score} \wedge \Delta_{bonus} \wedge p.score = 0 \wedge \neg\Delta_{score}[(o.bonus + 20) \div 2/o.score]$$

that is, after simplication:

$$\forall o : \text{Person}\,(o.bonus \in [0, 50]) \wedge \exists x : \text{Person}\,((x.bonus + 20) \div 2 \notin [0, 30])\ .$$

Interval propagation gives than if $x.bonus \in [0, 50]$, then $(x.bonus + 20) \div 2 \in [0, 35]$. The formula above is therefore satisfiable, which signals a *Domain Violation*.

5 Conclusion

In this paper, we propose to combine two approaches, to enhance the detection of inconsistencies in rule programs due to changes in ontologies.

$\mathcal{MDR}$ (for $\mathcal{M}$odel-$\mathcal{D}$etect-$\mathcal{R}$epair) is a pattern-based method to manage the impacts of ontology evolutions on rules defined over ontologies. Its "Detect" phase currently uses a syntactic approach. This approach consists in analysing the syntax tree of rules, to detect the rules impacted and the inconsistencies impacting them. We propose to complement this approach with a formal definition of inconsistencies in rule programs, which allows us to include a logic-based detection mechanism of these inconsistencies in $\mathcal{MDR}$.

Thanks to this combination, $\mathcal{MDR}$ can detect more cases of inconsistencies, on more complex rules, more reliably.

We have not implemented our proposal yet, however we are confident of its feasibility, since our formal approach is at the heart of the Rule Static Analysis features of the IBM ODM product [10], and $\mathcal{MDR}$ has been prototyped on top of IBM ODM. Furthermore, both have been deployed on large-scale, real-world use cases. $\mathcal{MDR}$ is experimented by Audi to verify that the tests elaborated

[2] BRMS stands for 'Business Rules Management System'.
[3] ODM stands for 'Operational Decision Manager'.

on seat belts conform to the European regulation [6], and by Hôpital Européen Georges-Pompidou in pharmaceutical validation of medication orders [7]. IBM ODM is used on a daily basis in mission-critical applications, such as the payment clearing and settlement platform of major credit card companies [9].

Acknowledgments The $\mathcal{MDR}$ approach has been developed in the context of the ONTORULE project at the Center for Advanced Studies (CAS) of IBM France. The formal approach has been developed as a joint project between the University of Freiburg and ILOG, now a part of IBM.

References

1. G. Antoniou, C.V. Damásio, B. Grosof, I. Horrocks, M. Kifer, J. Maluszynski, and P. F. Patel-Schneider. Combining rules and ontologies: a survey. Technical Report IST506779/Linköping/I3-D3/D/PU/a1, Linköping University, 2005. http://rewerse.net/publications/rewerse-description/REWERSE-DEL-2005-I3-D3.html.
2. B. Berstel and M. Leconte. Using constraints to verify properties of rule programs. In *Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, ICSTW'10, pages 349–354. IEEE Computer Society, 2010.
3. B. Berstel-Da Silva. Formalizing both refraction-based and sequential executions of production rule programs. In A. Bikakis and A. Giurca, editors, *Rules on the Web: Research and Applications*, volume 7438 of *Lecture Notes in Computer Science*, pages 47–61. Springer Berlin / Heidelberg, 2012.
4. B. Berstel-Da Silva. Verification of business rules programs. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg, Germany, 2012. http://www.freidok.uni-freiburg.de/volltexte/8799/.
5. A. Chniti, P. Albert, and J. Charlet. A loose coupling approach for combining OWL ontologies and business rules. In *RuleML2012@ECAI Challenge, at the 6th International Symposium on Rules Research Based and Industry Focused 2012*, volume 874, pages 103–110. CEUR Workshop proceedings, 2012.
6. A. Chniti, P. Albert, and J. Charlet. MDROntology : An ontology for managing ontology changes impacts on business rules. In *Joint Workshop on Knowledge Evolution and Ontology Dynamics 2012. In conjunction with International Semantic Web Conference (ISWC 2012).* CEUR Workshop proceedings, 2012.
7. A. Chniti, A. Boussadi, P. Degoulet, P. Albert, and J. Charlet. Pharmaceutical validation of medication orders using an OWL ontology and business rules. In *Joint Workshop on Semantic Technologies Applied to Biomedical Informatics and Individualized Medicine (SATBI+SWIM 2012). In conjunction with International Semantic Web Conference (ISWC 2012).* CEUR Workshop proceedings, 2012.
8. R. Djedidi and M.A. Aufaure. ONTO-EVO^{A}L ontology evolution approach guided by pattern modelling and quality evaluation. *Proceedings of the Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2010)*, 2010.
9. IBM. *VISA Europe: Processing payments with unprecedented agility and reliability*, 2011. http://www.ibm.com/software/success/cssdb.nsf/CS/SSAO-8FNJEB.
10. IBM. *IBM Operational Decision Manager v8.0 User's Manual*, 2012. http://publib.boulder.ibm.com/infocenter/dmanager/v8r0/.
11. ONTORULE Project. *Ontologies meet Business Rules*, 2012. http://ontorule-project.eu.