Towards the Verification of Ontologies with Rules

Joachim Baumeister, Thomas Kleemann, Dietmar Seipel
Institute of Computer Science, University of Würzburg, Germany
{baumeister,seipel}@informatik.uni-wuerzburg.de,
tomkl@kleemann-online.com

Abstract

The integration of a rule representation in ontology languages enhances the developer's abilities in the expression of knowledge. Likewise the integration creates new challenges for the design process of these knowledge bases. Thus, evaluation approaches have to cope with the merged methods. We introduce extensions to existing verification techniques to support the implementation of ontologies with rule enhancements, and we focus on the detection of anomalies that can especially occur due to the combined use of rules and ontological definitions.

Introduction

The use of ontologies as the building block of intelligent systems has shown its benefits in many applications. Currently, the expressiveness of ontology languages like OWL (Antoniou & van Harmelen 2004) is extended by a rule representation. One prominent representative is SWRL, i.e., the Semantic Web Rule Language (Horrocks *et al.* 2005), that is based on a Horn clause extension of OWL DL. With such an extension new evaluation issues arise, since the mixture of ontological definitions and rules can induce further anomalies.

This paper considers the verification of ontologies with rules, and presents an overview of possible anomalies integrating classical measures known from the verification of ontologies and the verification of rule bases. We motivate that besides the known anomalies further "mixed" anomalies can occur due to the combination of ontology definitions and rules. We distinguish the following classes of anomalies:

- *Circularity* in taxonomies and rule definitions.
- *Redundancy* due to duplicate or subsuming knowledge.
- *Inconsistency* because of contradicting definitions.
- *Deficiency* as a category comprising subtle issues describing questionable design in an ontology.

For the first three classes we sketch some anomalies that are likely to occur due to the mixture of rule-based and ontological knowledge; deficiency was considered in more detail, e.g., in (Baumeister & Seipel 2005; 2006).

Concerning the expressiveness of the ontology language we focus on the basic subset of OWL DL (which may easily transfer the described work to ontology languages other than OWL), and we mostly describe syntactic methods for the analysis of the considered ontology. In detail, we investigate the implications and problems that can be drawn from rule definitions in combination with some of the following ontological descriptions: 1. class relations like *subclass*, *complement*, *disjointness* 2. basic property characteristics like *transitivity*, *ranges and domains*, and *cardinality restrictions*. Furthermore, we focus on the basic elements of the rule language SWRL, i.e., Horn clauses with class or property descriptions as literals.

In the past, the verification of ontologies (mostly taxonomies) and rules (based on predicate logic), respectively, has been investigated separately. Thus, we build our work on top of the evaluation of taxonomic structures in ontologies (Gómez-Pérez 2001; Baumeister & Seipel 2005) as well as on top of classic work on the verification of rule bases, e.g., done by (Preece, Shinghal, & Batarekh 1992). The main contribution of our work is the extension of these measures by novel anomalies that are emerging from the combination of rule-based and ontological knowledge. Furthermore, we give a unifying view of all measures collected so far by introducing the so called *star of anomalies*. Of course, this collection of possible anomalies may always be incomplete, since additional elements of the ontology language may also introduce new possibilities of occurring anomalies.

The paper is organized as follows: We first define the basic notions that are necessary for the analysis of ontologies with rules. We further limit the expressiveness of the ontology language and rule language, respectively. After that, we present a collection of anomalies that are likely to occur due to the combined use of ontology definitions and rules. For each anomaly we also demonstrate a formal PROLOG implementation, instantly. We conclude the paper with a discussion of the presented work and give directions for future research.

Expressiveness and Basic Notions

For the analysis of ontologies with rules we restrict the range of considered constructs to a subset of OWL DL: we investigate the implications of rules that are mixed with subclass relations and/or the property characteristics transitivity, car-

dinality restrictions, complement, and disjointness.

For a class C and a property P to be used in rules, we call $C(x)$ a class atom and $P(x, y)$ a property atom. For the following it will be useful to extend the relations on classes and properties to relations on class and property atoms. Given two atoms A, A', we write $\odot(A, A')$, if both atoms have the same argument tuple, and their predicate symbols are related by $\odot$, i.e., if A and A' both are

- class atoms, such that $A = C(x)$, $A' = C'(x)$, and $\odot(C, C')$, or
- property atoms, such that $A = P(x, y)$, $A' = P'(x, y)$, and $\odot(P, P')$.

For example, the relation $\odot$ can be is-a , disjoint, complement, etc. Note, that from a relationship $\odot(A, A')$ it follows that A and A' are of the same type.

Implementation. The detection of anomalies has been implemented in SWI–PROLOG (Wielemaker 2003). Due to their compactness and conciseness we give the corresponding formal PROLOG definitions for the discussed anomalies.

Variables such as A, B, C, ..., A', or A_i can denote both class atoms and property atoms, whereas `As`, `Ais`, ..., denote sets of class atoms and property atoms. We denote a relation A is-a A' by `isa(A, A')`.

Rules $\beta \Rightarrow A$ are represented as terms `A-Ais`, where `Ais` $= [A_1, \ldots, A_n]$ is the list of body atoms (representing the conjunction $\beta = A_1 \wedge \ldots \wedge A_n$ and `A` is the head atom. They are stored in PROLOG facts of the form `rule(A-Ais)`. In PROLOG, disjunction (or) is denoted by "`;`", and negation is denoted by "`\+`". Since SWRL rules with conjunctive rule heads can be split into several rules, we can (without loss of generality) assume rule heads to be atomic, e.g., $(A \wedge B) \Rightarrow C \equiv A \Rightarrow C \wedge B \Rightarrow C$.

Complements and Disjointness of Classes

For classes there exists the construct *complementOf* to point to instances that do not belong to a specified class. The complement relation between a class `C1` and a class `C2` is denoted by `complement_of(C1,C2)` in PROLOG.

In OWL the disjointness between two classes is defined by the *disjointWith* constructor; with `disjoint(C1,C2)` we denote the disjointness between two classes `C1` and `C2`.

We call two classes `C1` and `C2` *incompatible*, if there exists a disjoint or (even) a complement relation between them.

```
incompatible(C1,C2) :-
   ( complement_of(C1,C2)
   ; disjoint(C1,C2) ).
```

Please note, that it is not necessary for the classes `C1`, `C2` to have a direct relation; such relations can be also transitively derived due to a taxonomic relations.

Taxonomic Relationships and Rules

Obviously, relationships B is-a A – where A and B are both class atoms or both property atoms with the same arguments – are equivalent to rules of the form $B \Rightarrow A$ with a single atom B in the body. Thus, we combine them into a single formalism in PROLOG:

```
derives(B, A) :-
   ( isa(B, A)
   ; rule(A-[B]) ).
```

We denote the transitive closure of `derives` by $\rightarrow$, and we compute it using the following standard scheme:

```
tc_derives(A, C) :-
   derives(A, C).
tc_derives(A, C) :-
   derives(A, B), tc_derives(B, C).
```

Categories of Anomalies

Figure 1 shows a hierarchical view of the anomalies with respect to their type (e.g., circularity, redundancy) and classified by the part of the knowledge base, where the anomaly was found. A *star of anomalies* can be defined that may grow with the expressiveness of the ontology and rule language, respectively. Here, we only present a selection of the depicted anomalies. Our work focuses on circularity, redundancy, and inconsistency, since deficiency has been considered in more detail, e.g., in (Baumeister & Seipel 2005).

Circularity

In general, we do not consider every circular definition as an anomaly. In rule bases partially circular constructs are commonly used for recursive definitions. However, many circular definitions (especially in the taxonomic part of the ontology) have severe implications to the reasoning capabilities of the underlying knowledge. In the past, *circular subclass definitions* have been already investigated by (Gómez-Pérez 2001), whereas *circular rule chains* were identified by (Preece & Shinghal 1994). For a combined use of rules and taxonomic knowledge we can identify the following additional anomalies.

Circularity between Rules and Taxonomy. A circularity is defined, if there exists a rule $A_1 \wedge \cdots \wedge A_n \Rightarrow A$, such that for some atom A_i from the antecedent it holds $A \rightarrow A_i$.

```
anomaly(circularity, A-Ais) :-
   member(Ai, Ais),
   tc_derives(A, Ai).
```

In such a case we can consider the specified rule as a restricted is-a relation between A and A_i. The anomaly may detect a misapplied taxonomic definition between the two concepts, which is similar to the *implication of superclasses* (see redundancy) but with an inverse is-a relation.

Circular Properties. We call properties circular if there exists a chain of properties linking a class/concept C via zero or more classes C_i to itself or one of its subsumees. Circular properties may lead to infinite models of the ontology under inspection. In purely description logic reasoners various blocking methods (Baader & Sattler 2001) ensure termination of the proof procedure in case of existentially quantified cycles. The combination with rules requires new methods, and decidability is not guaranteed in the general case.

Moreover the developer of an ontology might expect a fixpoint semantics for the cyclic definition (Georgieva & Maier

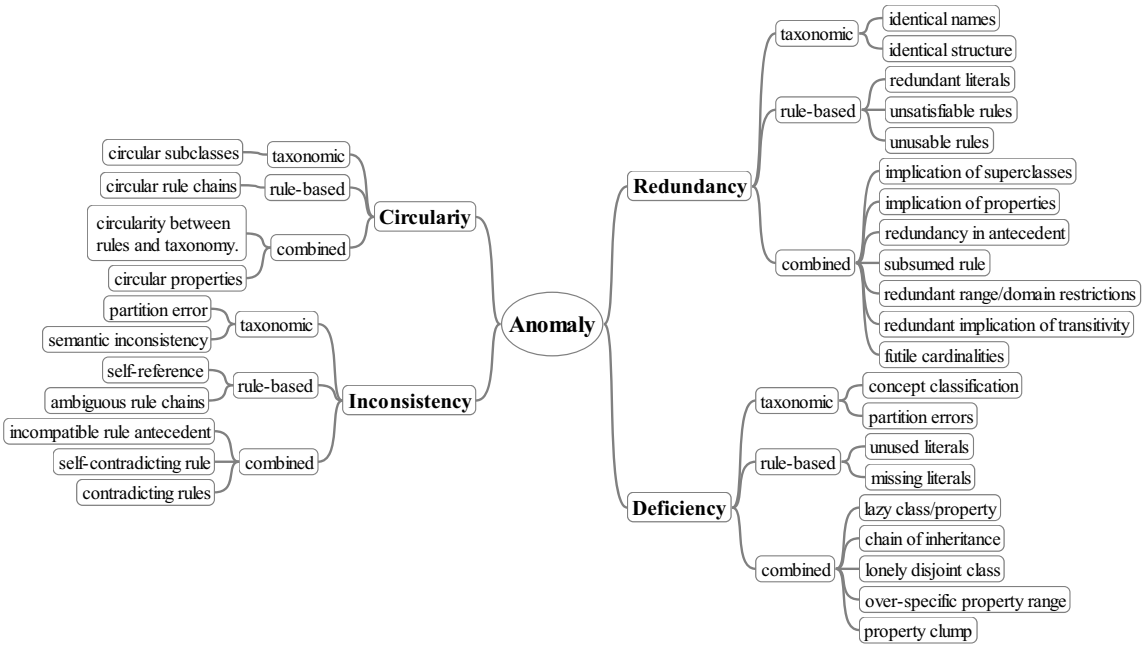

Figure 1: A star of anomalies for the combination of ontologies and rules.

2005). To the best of our knowledge that will not be provided by existing tools.

Natural sources of circularity are inverse and symmetric properties. Unlike these desired circularities a merging process of ontologies may lead to circular definitions. Combining ontologies with different views to properties may yield for example a class $vehicle$ that is connected by a property $isEquippedWith$ to a class $engine$, while in the merged ontology $engine$ was related to $vehicle$ by a property $isUsedIn$. The merging process is not completed until these properties are related to each other.

To detect circular properties we use the following predicates, where `property_restriction` is a generalization of restrictions like $someValuesFrom$, $allValuesFrom$, and applicable range restrictions on a property P.

```
anomaly(circular_property, C, Ps) :-
   tc_connected_classes(C, Ps, C),
   member(P, Ps), \+ symmetric(P).

tc_connected_classes(A, [P], C) :-
   connected_classes(A, P, C).
tc_connected_classes(A, [P|Ps], C) :-
   connected_classes(A, P, B),
   tc_connected_classes(B, Ps, C).

connected_classes(A, P, D) :-
   tc_derives(A, B),
   property_restriction(B, P, C),
   tc_derives(C, D).
```

The PROLOG call `anomaly(circular_property, C, Ps)` computes classes `C` that are part of a circular chain `Ps` of properties. The forming chain is computed using the predicate `tc_connected_classes`.

Inconsistency

Contradictory knowledge contained in ontological knowledge and rules often yields unintended and unexpected reasoning results. In the past, possible inconsistencies were investigated separately for both taxonomic knowledge (Gómez-Pérez 2001) and for rule-based knowledge (Preece & Shinghal 1994), respectively. Typical examples of inconsistencies are *contradicting rule* consequents for two rules with subsuming rule antecedent. For ontological knowledge the *partition error* is very common, i.e., a subclass of two or more classes that are contained in a disjoint partition (mutually disjoint classes). In the following, we only discuss inconsistencies that may occur due to the combined use of rules and ontology definitions.

Incompatible Rule Antecedent. For a rule $A_1 \wedge \cdots \wedge A_n \Rightarrow A$ there might exist an incompatibility relationship (disjointness or complementarity) between two body atoms A_i and A_j. Note that, according to our definitions this means that $A_i = C_i(x)$ and $A_j = C_j(x)$ are class atoms with the same argument x, and that C_i and C_j are disjoint or complements. A rule having incompatible concepts may be also considered as a redundancy, since the rule would never fire.

```
anomaly(incompatible_antecedent, A-Ais) :-
   subset([B, C], Ais),
   incompatible(B, C).
```

However, from our point of view a redundancy should be reported when it is likely that the detected part should be removed from the knowledge base. Here, we think that this

anomaly is more likely to be the result of a defective alignment of concepts.

Self–Contradicting Rule. For rule $A_1 \wedge \cdots \wedge A_n \Rightarrow A$ there might exist a complementOf or a disjointWith relationship between A and one of its body atoms A_i. Note that, according to our definitions this means that $A = C(x)$ and $A_i = C_i(x)$ are class atoms with the same argument x, and that C and C_i are disjoint or complements.

```
anomaly(contradicting_consequent, A-Ais) :-
   member(Ai, Ais),
   incompatible(A, Ai).
```

If such a rule would fire, then its consequent A would contradict the precondition A_i.

Contradicting Rules. Two rules $r = \beta \Rightarrow A$ and $r' = \beta' \Rightarrow A$' are contradicting, if the conjunction β subsumes β' with respect to $\rightarrow$, but A and A' are contradicting. If r' would fire, then also the stronger rule r would fire. Consequently, the conclusions A' and A would be contradicting.

```
anomaly(contradicting_rules, R1, R2) :-
   R1 = A1-Bs1, R2 = A2-Bs2,
   subsumes(Bs1, Bs2),
   incompatible(A1, A2).
```

A conjunction $\beta = A_1 \wedge \ldots \wedge A_n$ subsumes another conjunction $\beta' = B_1 \wedge \ldots \wedge B_m$ with respect to the relation $\rightarrow$, if there exists an instance $\beta\theta$ of β and a mapping $\pi : \{1, \ldots, n\} \mapsto \{1, \ldots, m\}$, such that $\forall\ 1 \leq i \leq n : B_{\pi(i)} \rightarrow A_i\theta$. The following predicate computes the substitution θ and the mapping π:

```
subsumes(As, Bs) :-
   copy_term(Bs, Cs),
   subsumes_(As, Bs),
   variant(Bs, Cs).

subsumes_([A|As], Bs) :-
   member(B, Bs), tc_derives(B, A),
   subsumes_(As, Bs).
subsumes_([], _).
```

Since the application of the predicate `tc_derives` to two atoms `A` and `B` will unify their argument vectors instead of just finding an instance $A\theta$ of A, such that $A\theta \rightarrow B$ (mapping), we have to test that the value of `Bs` after the call `subsumes_(As, Bs)` is a variant of the value before that call, which was safed in `Cs`. The consequents $A = C(x)$ and $A' = C'(x')$ are contradicting, if the corresponding classes C and C' are disjoint or complements.

We can generalize the described anomaly as follows: There are two (not necessarily disjoint) sets of rules that are deriving two semantically contradicting conclusions. However, this type of anomaly cannot be detected in a purely syntactic manner.

Redundancy

We define redundant knowledge as ontological definitions or rules that can be removed from the knowledge base without changing the intended semantics. Such redundancies can be clearly identified when available in the knowledge base. Many types of redundancy were separately discussed for ontologies in (Gómez-Pérez 2001), e.g., identical concepts, and for rules in (Preece, Shinghal, & Batarekh 1992), e.g., subsuming rules. In the context of this paper we introduce redundancies that can occur due to the combined use of ontological definitions and rules.

Implication of Superclasses. If A, A_i are either class or property atoms, then a rule $r = A_1 \wedge \cdots \wedge A_n \Rightarrow A$, such that $A_i \rightarrow A$ for some A_i, is redundant.

```
anomaly(implication_of_superclasses, R) :-
   R = A-Ais,
   member(Ai, Ais),
   tc_derives(Ai, A).
```

Here, classes are only subsuming under certain conditions that are given in the rule condition, i.e., an incorrect assignment of the subclass relation may exist. If $A_i \equiv A$, then the equivalence may be incorrectly assigned, since the rule condition denotes a restriction on the implication. This can be seen as a special case of rule subsumption, since the fact $A_i \rightarrow A$ can be seen as a rule $A_i \Rightarrow A$, which subsumes the rule r given above.

Redundancy in the Antecedent of a Rule. If for two atoms A_i, A_j in the antecedent of a rule $A_1 \wedge \cdots \wedge A_n \Rightarrow A$ we have $A_i \rightarrow A_j$, then the atom A_j is redundant, and it can be removed.

```
anomaly(redundancy_in_antecedent, A-Ais) :-
   subset([Ai, Aj], Ais),
   tc_derives(Ai, Aj).
```

As a special case, this form of redundancy can occur for the equivalence $A_i \equiv A_j$. This anomaly may alternatively point to an incorrect mapping between the elements A_i and A_j, when these two elements were aligned from different ontologies.

Subsumed Rule. A rule $r = A_1 \wedge \ldots \wedge A_n \Rightarrow A$ subsumes a rule $r' = B_1 \wedge \ldots \wedge B_m \Rightarrow B$ if

$$\neg A_1 \vee \ldots \vee \neg A_n \vee A \sqsubseteq_{\rightarrow} \neg B_1 \vee \ldots \vee \neg B_m \vee B.$$

This means that there exists an instance $r\theta$ of r and a mapping $\pi : \{1, \ldots, n\} \mapsto \{1, \ldots, m\}$, such that

$$A\theta \rightarrow B \text{ and } \forall\ 1 \leq i \leq n : \neg A_i\theta \rightarrow \neg B_{\pi(i)}.$$

For the positive atoms this means that $B_{\pi(i)} \rightarrow A_i\theta$. In this case the rule r' may be omitted without any changes to the ontology as a whole.

```
anomaly(subsumed_rule, A-As, B-Bs) :-
   subsumes_rule(A-As, B-Bs).

subsumes_rule(A-As, B-Bs) :-
   copy_term(Bs, Cs),
   tc_derives(A, B),
   subsumes_(As, Bs),
   variant(Bs, Cs).
```

Redundant Implication of Transitivity. A rule $r = P(x,y) \wedge P(y,z) \wedge \beta \Rightarrow P(x,z)$ contains a redundant definition of the property P when P is already defined to be transitive. Then, $P(x,z)$ can be already derived by the OWL reasoner.

Often such a redundancy can be explained by an erroneous assumption of the transitivity during an ontology integration process, since the rule defines a more restrictive condition of transitivity, if the conjunction β is not empty. For this reason, the anomaly may be also classified as an inconsistency due to an incorrect mapping.

```
anomaly(redundant_transitivity, Rule) :-
   Rule = P_xz-Body,
   P_xz =.. [P, X, Z],
   P_xy =.. [P, X, Y], P_yz =.. [P, Y, Z],
   subset_non_ground([P_xy, P_yz], Body).
```

This is also a special case of rule subsumption, since the transitivity of a property P can be expressed as a rule $P(x,y) \wedge P(y,z) \Rightarrow P(x,z)$, which subsumes the rule r given above.

Redundant Range and Domain Restrictions. Range and domain restrictions are a global way to define constraints on the arguments of properties. These restrictions may interfere with rules. To integrate the analysis of range and domain restrictions we interpret a domain restriction of class D associated with a property P as a PROLOG rule `D(X) :- P(X,Y).` Likewise a range restriction of class R will look like `R(Y) :- P(X,Y).` Because properties may have multiple domain and range restrictions, multiple rules may be generated.

Given this rewriting, we can use the existing predicates on redundant or conflicting rules to evaluate the ontology with respect to redundancies and conflicts of domain and range restrictions.

In addition to the interference with rules, domain and range restrictions are prone to redundant restrictions in property hierarchies. A subproperty $P' \sqsubseteq P$ has to comply with the restrictions of its subsumer P. A restriction of a subproperty is only useful if the restriction of the subproperty is more restrictive than that of its subsumer.

```
anomaly(redundant_range, Q, S) :-
   tc_derives(Q, P),
   hasRange(Q, S), hasRange(P, R),
   subsumes(S, R).


anomaly(redundant_domain, Q, S) :-
   tc_derives(Q, P),
   hasDomain(Q, S), hasDomain(P, R),
   subsumes(S, R).
```

For those domain and range restrictions of subproperties that are more general we define a warning rule. The developer should be warned of all properties P, P' with $P' \sqsubseteq P$ and $range(P) \sqsubseteq range(P')$ or $domain(P) \sqsubseteq domain(P')$ respectively.

Futile Cardinalities. Current ontology editors support the creation of cardinality restrictions with few clicks of a mouse. Not all of the created restrictions are useful. Especially those with cardinalities 0 or 1 are often superficious or should be replaced by a property axiom without cardinalities.

Mincardinality 0. A cardinality restriction like ≥ 0 is trivially true for all individuals because every individual is linked to zero or more individuals via any property. Thus, an OWL restriction like the following may be omitted.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#P" />
  <owl:minCardinality
    rdf:datatype="&xsd;nonNegativeInteger">
    0</owl:minCardinality>
</owl:Restriction>
```

To detect and reference these anomalies we define a rule for the predicate anomaly:

```
anomaly(redundant_mincardinality_0, C) :-
   min_cardinality_restriction(C, P, 0).
```

Every class C with a restriction of `minCardinality 0` is detected and returned.

Maxcardinality 0. Classes with restrictions on a property P of `maxCardinality 0`enforce the absence of any individual linked via P with the members of this class. This is equivalent to a restriction of `allValuesFrom #Nothing`. We assume `#Nothing` to be the empty class (bottom concept).

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#P" />
  <owl:maxCardinality
    rdf:datatype="&xsd;nonNegativeInteger">
    0</owl:maxCardinality>
</owl:Restriction>
```

This OWL fragment may be replaced by the following restriction:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#P" />
  <owl:allValuesFrom rdf:resource="#Nothing" />
</owl:Restriction>
```

To indicate the presence of these maximum cardinality anomalies we use the predicate:

```
anomaly(maxcardinality_0, C) :-
   max_cardinality_restriction(C, P, 0).
```

Mincardinality 1. Restrictions of `minCardinality 1` on a property P enforce at least one individual to be linked by P. They have exactly the same semantics as the `someValuesFrom` restriction into the `#Thing` class (top concept) or into the range of P.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#P" />
  <owl:minCardinality
    rdf:datatype="&xsd;nonNegativeInteger">
    1</owl:minCardinality>
</owl:Restriction>
```

This OWL fragment may be replaced by a restriction without cardinalities:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#P" />
  <owl:someValuesFrom rdf:resource="#Thing" />
</owl:Restriction>
```

In order to use the anomaly predicates in rules we may also add an equivalent rule `P(X,f(X)) :- C(X).` Restrictions on $f(X)$ to `#Thing` may be omitted, because all individuals are members of `#Thing`. The newly introduced function f is not valid in SWRL but compatible with our anomaly predicate. To indicate the presence of cardinalities, that can be replaced by a restriction without cardinalities, we use the following predicate:

```
anomaly(mincardinality_1, C) :-
   min_cardinality_restriction(C, P, 1).
```

Maxcardinality 1. Restrictions of $\leq 1\ P$ allow for at most one individual to be linked by property P. This cardinality is not needed if P is a functional property or a sub–property of a functional property. Functionality of P enforces the uniqueness globally; so a local restriction is subsumed by the global one. To indicate the presence of cardinalities, that can be replaced by a restriction without cardinalities we use the following predicate:

```
anomaly(subsumed_maxcardinality_1, C) :-
   functional_property(P),
   max_cardinality_restriction(C, P, 1).

anomaly(subsumed_maxcardinality_1, C) :-
   functional_property(Q),
   tc_derives(P, Q),
   max_cardinality_restriction(C, P, 1).
```

We may also add an equivalent rule

```
sameAs(Y,Z) :- C(X), P(X,Y), P(X,Z)
```

to use our anomaly predicates on rules.

Discussion

Ontologies as the building block for intelligent applications are currently extended by a rule representation layer, e.g., SWRL/RuleML as a prominent example. We have motivated that with the increased expressiveness of such a knowledge representation language also new evaluation issues arise. The presented work contributed a combined overview of verification methods detecting anomalies that may occur in ontological constructs, in the rule part, but also due to the combined use of ontological constructs and rules. All anomalies were also described by a corresponding PROLOG implementation, which was also used in the actual application for *anomaly detection*. In (Baumeister & Seipel 2006) a combined view was also given, but with a strong focus on deficiencies and the application of refactoring methods, i.e., the safe elimination of the detected anomalies.

At the moment, the implementation is tested on small example ontologies. A case study with a real world ontology is still pending. In such a study it would be interesting to see the frequencies in which the particular anomalies occur (however, some of the described anomalies were actually motivated by the experience we have made when building larger rule-based systems with a small ontology layer). As stated in the introduction, only a small fraction of the possible ontologies was introduced. A trustable verification tool should include tests for a more comprehensive set of anomalies. Therefore, future work should produce an exhaustive collection of anomalies. Here, anomalies should be classified not only by *type* (e.g., redundancy) and *occurs-in* (e.g., rule-based vs. combined), but also by the underlying *expressiveness*. For example, we expect ontologies with the expressiveness of OWL LITE to have a subset of possible anomalies when compared to ontologies with the expressiveness of OWL DL.

References

Antoniou, G., and van Harmelen, F. 2004. *Web Ontology Language: OWL*. Springer. chapter 4, 67–92. In Staab & Studer: Handbook of Ontologies.

Baader, F., and Sattler, U. 2001. An Overview of Tableau Algorithms for Description Logics. *Studia Logica* 69:5–40.

Baumeister, J., and Seipel, D. 2005. Smelly Owls – Design Anomalies in Ontologies. In *Proc. of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 215–220. AAAI Press.

Baumeister, J., and Seipel, D. 2006. Verification and Refactoring of Ontologies With Rules. In *Managing Knowledge in a World of Networks: 15th International Conference, EKAW, LNAI 4248*, 82–95. Springer.

Georgieva, L., and Maier, P. 2005. Description Logics for Shape Analysis. In Aichernig, B. K., and Beckert, B., eds., *SEFM*, 321–331. IEEE Computer Society.

Gómez-Pérez, A. 2001. Evaluation of Ontologies. *International Journal of Intelligent Systems* 16(3):391–409.

Horrocks, I.; Patel-Schneider, P. F.; Bechhofer, S.; and Tsarkov, D. 2005. OWL Rules: A Proposal and Prototype Implementation. *Journal of Web Semantics* 3(1):23–40.

Preece, A., and Shinghal, R. 1994. Foundation and Application of Knowledge Base Verification. *International Journal of Intelligent Systems* 9:683–702.

Preece, A.; Shinghal, R.; and Batarekh, A. 1992. Principles and Practice in Verifying Rule-Based Systems. *The Knowledge Engineering Review* 7 (2):115–141.

Wielemaker, J. 2003. An Overview of the SWI-Prolog Programming Environment. In *Proc. 13th Int. Workshop on Logic Programming Environments*, 1–16. Heverlee, Belgium: Katholieke Universiteit Leuven. CW 371.