

Verification and Refactoring of Ontologies with Rules



Joachim Baumeister and Dietmar Seipel
University of Würzburg, Germany

EKAW 2006 - Podebrady, CZ

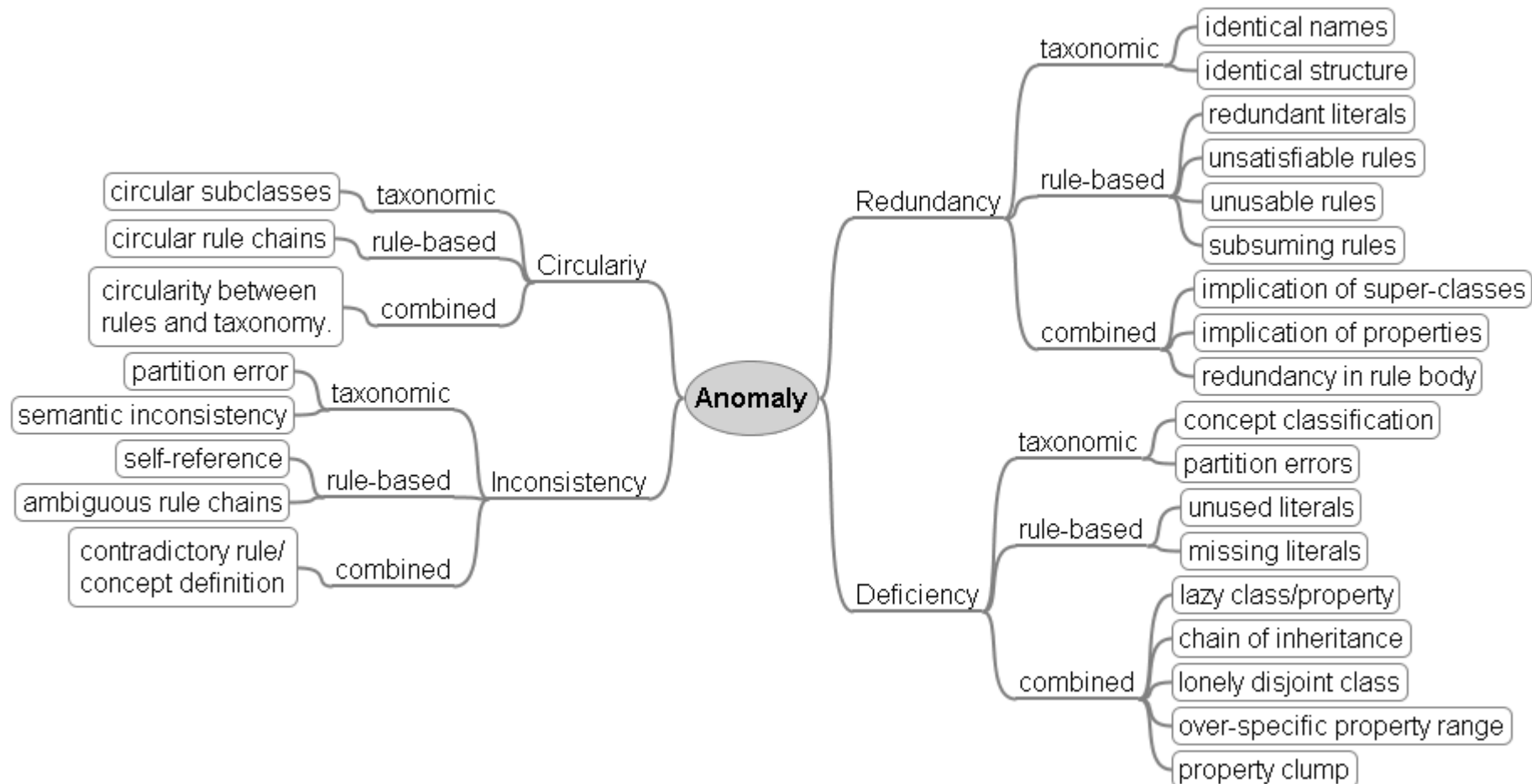
Use of Rules in Ontologies

- Besides many proprietary approaches the SWRL proposal tries to define a standardized representation of rules within OWL
- For many applications the use of rules in ontologies are helpful or essential, e.g.,
 - business rules for semantic web services
 - knowledge intensive applications (e.g., in medicine)
 - augmenting existing rule applications with ontologies
- Due to the integration of ontologies with rules some evaluation issues arise!

Evaluation Issues

- Large body of verification research for both
 - ontologies (e.g., A. Gómez-Pérez)
 - rules (e.g., A. Preece)
- With the intermixture of ontological knowledge and rules new verification measures necessary
 - e.g., trivial redundancy: $A \Rightarrow B$ / $A \text{ is-a } B$
- Methods: *Verification* and *Refactoring*
 - *Verification*: syntactic analysis for the detection of anomalies (e.g., redundancy, deficiency)
 - *Refactoring*: elimination of an anomaly without changing the intended semantics of the ontology; aiming for a better design of the ontology

A Star of Anomalies (incomplete)



Declarative Methods

- Use Prolog to declaratively define the measures
 - simply to define and instantly executable
(engine based on SWI-Prolog incl. Semantic Web library)
- Syntax and Notions
 - for rules $Body \Rightarrow Head$: **Head-Body**
(**Head** to be atomic)
 - General elements:

```
element(A) :-  
    ( class(A); property(A) ).
```
 - Some meta-predicates like

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

Notions (cont.)

- Derivation of concepts/properties:

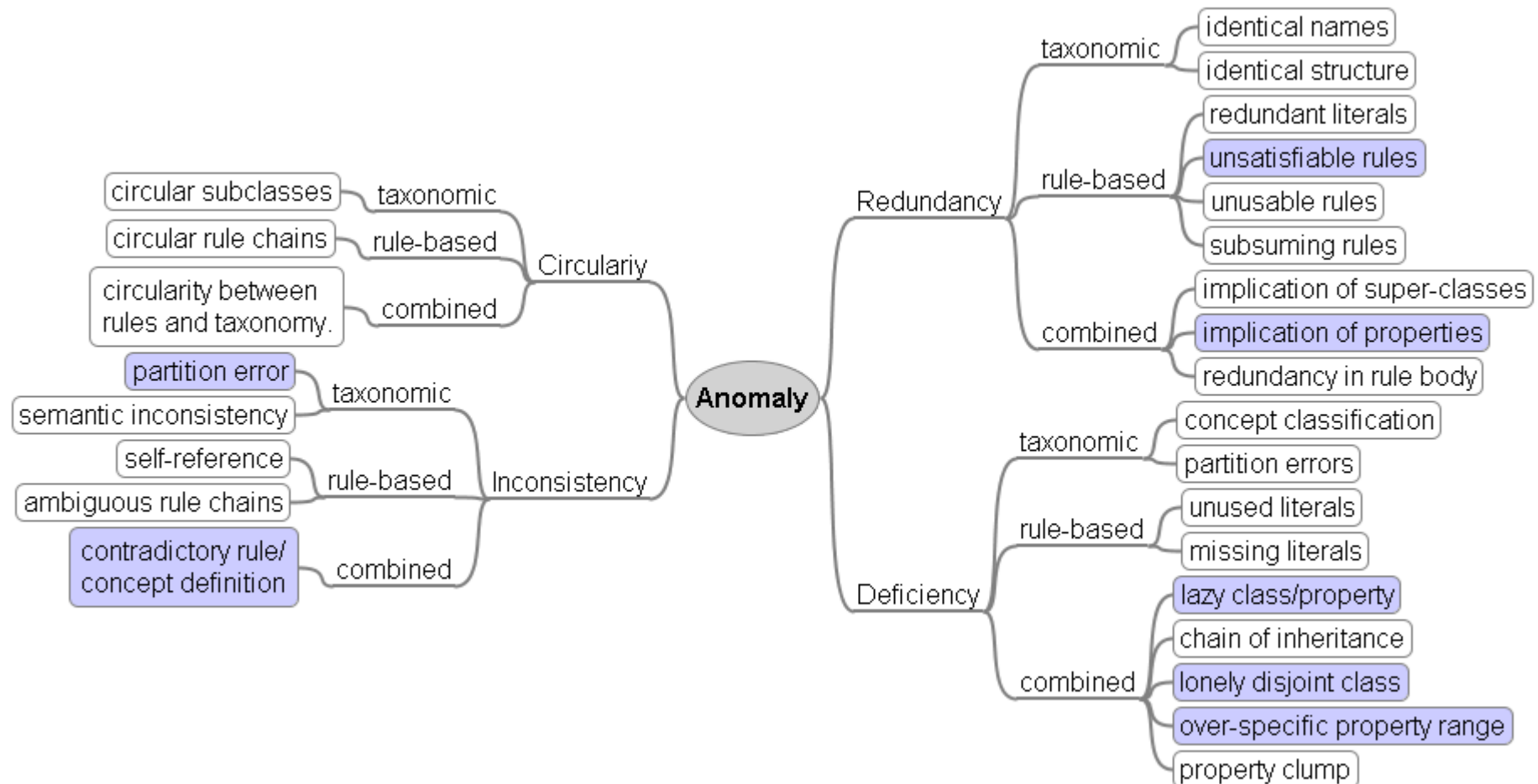
```
derives(B, A) :-  
  ( isa(B, A)  
    ; rule(A-[B]) ).
```

- Transitive closure of arbitrary predicate <P>:

```
tc_<P>(A, C) :-  
  ( <P>(A, C)  
    ; <P>(A, B), tc_<P>(B, C) ).
```

... used for predicates `is-a` and `derives`

Selected Examples



Examples for Redundancy

Redundant Implication of Transitivity

$P(x, y) \wedge P(y, z) \wedge \beta \Rightarrow P(x, z)$ - P is a *transitive property*

- transitivity already derivable by OWL reasoner
- no real redundancy, since rule defines more restrictive condition for transitivity with β (for empty β : *strict redundancy*)

Redundancy in Antecedent of a Rule

$A_1 \wedge \dots \wedge A_n \Rightarrow A$ - but $A_i \rightarrow^* A_k$

```
anomaly(redundancy_in_antecedent, A-Body) :-
    tc_derives(Ai, Aj),
    member(Ai, Body), member(Aj, Body).
```

- special case: $A_i \equiv A_k$
- may point to an incorrect mapping of A_i and A_k

Unsatisfiable Rule Condition

- At least one literal neither unifies with input literal (e.g. instances of ontological concepts) nor consequent of another rule

```
anomaly(unsatisfiable_condition, _-Body) :-  
    member(A, Body),  
    \+ fact(A),  
    \+ rule(A-_).
```

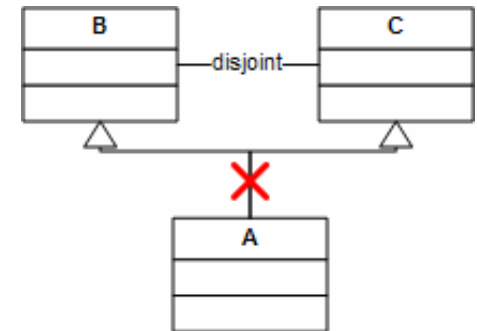
- With the contradictory use of complementOf / disjoint

```
anomaly(unsatisfiable_condition, _-Body) :-  
    member(A, Body), member(B, Body),  
    incompatible(A, B).
```

Examples for Inconsistency

- **Partition Error:** For classes/instances A, B, C:

```
anomaly(partition_error, A-[B, C]) :-
    disjoint(B, C),
    isa(A, B), isa(A, C).
```



- **Self-Contradicting Rule:** $A_1 \wedge A_2 \wedge \dots \wedge A_n \Rightarrow A$

where $A=C(x)$ and $A_i=C_i(x)$;

\Rightarrow C and at least one C_i are incompatible

```
anomaly(contradicting_rule_consequent, A-Body) :-
    member(B, Body), incompatible(A, B).
```

Deficiency

- More subtle measures: neither contradicting nor redundant knowledge, but uncovers possible bad design of ontology
 - Relations to Software Engineering (bad smells) and database schema design (anomalies)
- Refactoring: eliminating the deficiency but retaining the intended semantics
 - should yield better maintainability of the ontology
- Following measures are only *heuristics* for identifying deficiencies

Lazy Class/Property

- An element may be lazy if all/many of the following indicators occur
 - element is a leaf in the hierarchy
 - no rules use this element (probably tolerating few rules)
 - no instances of this element
- Reasons for laziness:
 - merge of ontologies include many unused elements
 - element evolution to more special/general elements

```
anomaly(lazy_element, A):-
```

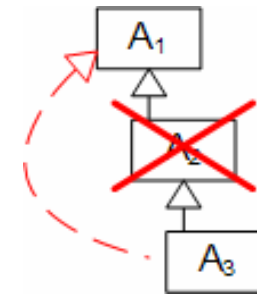
```
  element(A),  
  \+ isa(_, A),  
  \+ in_rule(A),  
  \+ instance(_, A).
```

```
in_rule(A) :-  
  rule(Head-Body),  
  ( A = Head  
  ; member(A, Body) ).
```

Refactoring: Delete Term

Removing unused elements should be considered with reasonable care:

1. Reconnect hierarchy
2. Reattach attributions of the term (e.g., transitivity for lazy property)
3. Edit rule including term: $A_1 \wedge A_2 \wedge \dots \wedge A_n \Rightarrow A$
 1. term in rule consequent: remove rule
 2. term in rule antecedent: remove rule (default) or change rule by removing only literal (semi-automatic setting)



For changing rules (step 3.2): consider the creation of anomalies, i.e., redundant or ambivalent rules!

Lonely Disjoint Class

- Class that is not disjoint with its siblings but has disjoint relations to a set of classes that are siblings

```
anomaly(lonely_disjoint, C) :-  
    siblings(Cs),  
    disjoint_partition([C|Cs]),  
    \+ ( sibling(C, M), disjoint(C, M) ).
```

- Reasons:
 - Integration of ontologies
 - Manual modification without removing old disjoint relation
- Refactoring:
Quite simple: deletion of disjoint statements

Over-Specific Property Range

- Common for manual development, where initially ranges of properties tend to be too specific
- E.g.: $R_{temp} = \{\text{very high, high, normal, low, very low}\}$.
 - $\{\text{very high, high}\}$ and $\{\text{low, very low}\}$ in equivalent use, e.g., for rules antecedents:

```
anomaly(over_specific, R1, R2, has_value(P, [V1, V2])):-
    R1 = A-Body1, R2 = A-Body2, R1 \= R2,
    delete(has_value(P, V1), Body1, B),
    delete(has_value(P, V2), Body2, B).
```

- Refactoring: for rules and hasValue restrictions the original values are replaced with aggregated ones using a mapping table:

v	very high	high	normal	low	very low
$M(v)$	high	high	normal	low	low

Check for redundancy (probable) & inconsistency (unwanted)!

Discussion

- Ontology languages are currently extended by a rule notion (e.g. SWRL as a proposal for an OWL extension)
- Combination of ontology knowledge with rules induce new evaluation issues (→ mostly syntactic analysis of ontology)
 - Revisiting existing evaluation measures
 - Thinking about new measures that are likely to occur
- Extend the classic work by more subtle measures considering the design of the ontology
 - Use of refactoring for a lossless compression of the ontology
- Presented work is only a start into this direction
 - more comprehensive collection of syntactic measures
 - existential & universal quantification? cardinalities?

Verification and Refactoring of Ontologies with Rules



Joachim Baumeister and Dietmar Seipel
University of Würzburg, Germany

EKAW 2006 - Podebrady, CZ