Bounds: Expresssing Reservations about Incoming Data

Martin G. Skjæveland[1] and Audun Stolpe[2]

[1] Department of Informatics, University of Oslo, Norway
martige@ifi.uio.no
[2] Norwegian Defence Research Establishment
Audun.Stolpe@ffi.no

Abstract. This position paper presents generic constraints useful for expressing dependencies between source and target datasets in situations where the source is to be merged into the target and the target is particular with what data it is willing to accept. These constraints, which we call *bounds*, are based only on the graph-like triple structure of RDF graphs and draws ideas from of bisimulation from modal logic [1], conservative extensions of ontologies [3, 5], and relational peer database exchange [2]. Although simple and generic, bounds still have powerful and practical natural interpretations, and favorable computational properties. An RDF vocabulary for expressing, publishing, and re-using bounds is available, together with an evaluated prototype implementation for checking such constraints.

1 Bounds

Let U, B and L respectively denote pairwise disjoint, fixed and infinite sets of *URIs*, *blank nodes* and *literals*. Fix $\mathcal{U} = U \cup B \cup L$ as the set of *elements*. Define the set of *(RDF) triples* as the set $\mathcal{T} = (U \cup B) \times U \times \mathcal{U}$. A triple is commonly written as a sequence of its elements, $\boldsymbol{t} = \langle s, p, o \rangle$, where s, p and o are called respectively the *subject*, *predicate* and *object* of the triple. An *(RDF) graph* G is a finite set of triples. If G is a graph, then $\mathcal{U}(G)$ is the set of elements occurring in G.

Definition 1 (Homomorphism). *Let G and H be graphs. A homomorphism $h : G \to H$ is a function $h : \mathcal{U}(G) \to \mathcal{U}(H)$, which maps elements in G identically $h(u) = u$ for all $u \in \mathcal{U}(G)$, and satisfies the homomorphism condition; for all $s, p, o \in \mathcal{U}$:*

$$\langle s, p, o \rangle \in G \quad \Rightarrow \quad \langle h(s), h(p), h(o) \rangle \in H.$$

Definition 2 (Bounded Homomorphism). *Let $h : G \to H$ be a homomorphism, then a simple* bound *is one of following conditions; for all $s, p, o \in \mathcal{U}$:*

$$\langle h(s), p, o \rangle \in H \quad \Rightarrow \quad \langle s, p, o \rangle \in G \qquad (\mathsf{S})$$
$$\langle s, h(p), o \rangle \in H \quad \Rightarrow \quad \langle s, p, o \rangle \in G \qquad (\mathsf{P})$$
$$\langle s, p, h(o) \rangle \in H \quad \Rightarrow \quad \langle s, p, o \rangle \in G \qquad (\mathsf{O})$$
$$\langle s, p, o \rangle \in H \quad \Rightarrow \quad \langle s, p, o \rangle \in G \qquad (\top)$$

New bounds may be built from the simple bounds by combining them conjunctively and/or disjunctively. A bounded homomorphism *is a homomorphism that satisfies a bound. If h satisfies the bound β, we call h a β-map.*

Bounded homomorphisms provide detailed control over the process of merging a source into a target graph by interlocking the two graphs in a reciprocal simulation of varying degrees of strength. Simplified, the homomorphism condition ensures that source data is transferred to the target in a structure-preserving way, whereas the different bounds prevent the incoming data from interfering with the data that the recipient already possesses. Thus, bounded homomorphisms are a kind of structural, closed-world restrictions for expressing reservations about incoming data to a target RDF graph, based on what is already in the target.

There are 19 different non-equivalent bounds for homomorphisms. If we let '$\bot$' designate a homomorphism without bound, we can arrange all 19 bounds and $\bot$ in a lattice according to logical implication; this is done in Figure 1. Here, if we have $\beta_1 \leq \beta_2$ for two bounds β_1, β_2, it means that β_2 is as least as strong as β_1, such that any β_2-map is also a β_1-map. The weakest bounded homomorphism is the $(\mathsf{S} \wedge \mathsf{P} \wedge \mathsf{O})$-map, while $\top$-map is the strongest.

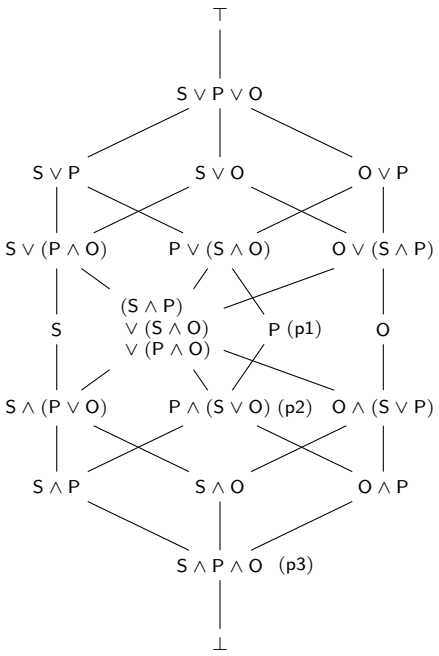

Fig. 1. Bounds

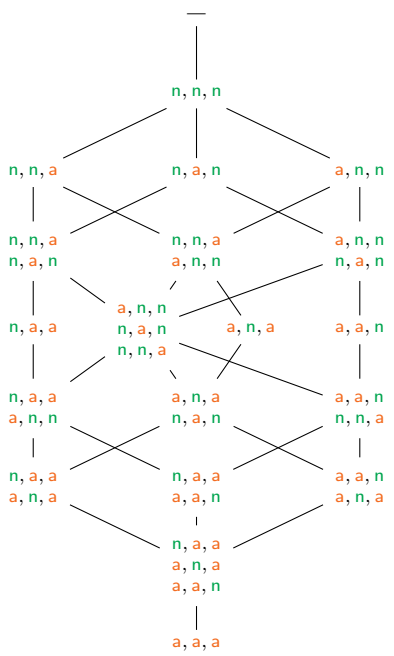

Fig. 2. Bounds, permissible triple patterns.

Many of the restriction that the bounds pose have rather natural and intuitive interpretations, and may be used to exercise different levels of control depending on the intended semantics of the data to be consumed by the target. Figure 2 offers a compact explanation of the patterns of new triples that a target T is willing to accept from a source S under the different bounds, i.e., the permissible triples in $T \setminus S$ after the exchange. In the figure,

the patterns use n ('new') to indicate that an element in this position *must* be new to T ($\mathsf{n} \notin T$), while a ('any') specifies that *any* element is allowed ($\mathsf{a} \in \mathcal{U}$). Multiple patterns in a position of the lattice mean that a triple matching any of the patterns satisfies the corresponding bound. We will now explain a few of these patterns to give a flavor of what bounds are capable of expressing.

- A $\top$-map allows no triples to be added to target—the target is 'write-protected'.
- A $\bot$-map allows any triple to be added to the target.
- The weakest bounded map, the $(\mathsf{S} \wedge \mathsf{P} \wedge \mathsf{O})$-map, allows the target to consume triples where at least one element does not already occur in the target. This bound is similar to *conservative extensions* in the sense that it does not allow novel assertions to be added to the protected RDF graph if they are solely about known entitites and/or concepts [3, 5]. That is, no new information expressed solely in the language of the target may be added to the target, i.e, the target does no accept triples which (only) re-arrange elements already in the target.
- A $(\mathsf{S} \vee \mathsf{P} \vee \mathsf{O})$-map requires all elements of added triples to be new to the target.
- A P-map does not accept adding new triples where the predicate already exists in the target. If, on the other hand, the predicate is new to the target, the triple is accepted.

We can also limit the applicability, or the scope, of a bound to certain values, e.g., we can require that a P-map only applies to `rdf:type` predicates, thus effectively 'sealing' of all membership axioms in the target. Similarly, we can specify that a S-map only applies to triples where the predicate is `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` or `rdfs:range`, hence disallowing new superclasses, superproperties, and domain and range definitions for resources in the target, protecting the target against what is known as *ontology hijacking* [4].

2 Vocabulary

We have designed an RDF vocabulary for expressing bounds called 'Boundz': `http://sws.ifi.uio.no/vocab/boundz`. The vocabulary comprises 32 classes, 34 properties and 3 individuals; an informal and simplified overview of its most important top-level classes and properties is depicted in Figure 3.

The vocabulary is designed to allow datasets to protect themselves against disruptive or potentially harmful patterns of incoming data. It is also meant to promote re-use of typical bound sets, anticipating use-cases where a vocabulary is published along with a set of restrictions that protects it from, say, ontology hijacking.

In general though, a bound-set is typically used to control the exchange of data between several parties by listing the contributors and recipients of the exchange together with the bounds that each exchange is required to satisfy. Every exchange has a payload, which is the set of conformant triples according to the relevant bounds. That is, the payload of an exchange is the set of triples that do not violate any of the bounds that are placed as requirements on that exchange.

In order to negotiate an acceptable payload, bounds themselves allow for exceptions and restrictions. Restrictions narrow the scope of a bound. For instance, a bound may be

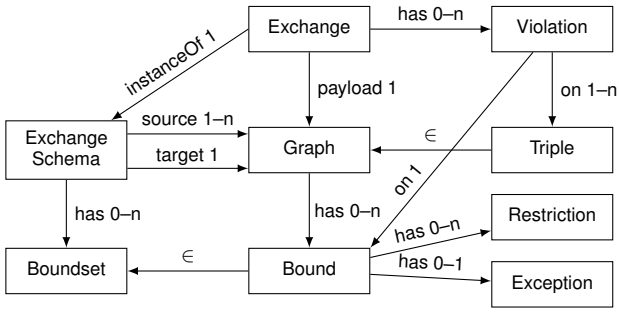

Fig. 3. Boundz vocabulary, an informal and simplified excerpt.

scoped so as to apply only to triples with `rdf:type` as predicate, as exemplified above. Other possible scopes comprise an entire namespace, or all resources of a given type.

An exception, on the other hand, prescribes how a violation is to be handled: shall the exchange be aborted altogether, or shall only the problematic triples be ignored.

The listing below shows an example of a specification of an exchange; we refer the reader to the vocabulary documentation for an explanation of the terms.

```
ex:test a bz:ExchangeSchema ;
 bz:hasSource ex:source1, ex:source2; bz:hasTarget ex:target;
 bz:targetReasoning false; bz:sourceReasoning  false;
 bz:outputPayload   false; bz:outputViolations true;
 bz:hasBound
  [a bz:KKspo; bz:hasException bz:ignoreViolations],
  [a bz:Aso; bz:predicateValue rdf:type, rdfs:subClassOf],
  [a bz:s; bz:predicateNamespace rdfs:; bz:hasException bz:ignoreSource],
  [a bz:T; bz:subjectClass foo:Bar], [a bz:T; bz:objectClass foo:Bar] .
```

3 Implementation

The complexity of deciding if a bounded homomorphism exists between two graphs is polynomial in the size of the two graphs [6]. Since bounds use triple patterns, they may be represented using SPARQL queries, however, intitial experiments tell us that a simple direct algorithm which loops through all possible matches for applicable triples patterns is more efficient. A prototype implementation with test evaluation is available at `http://sws.ifi.uio.no/project/boundz/`. The implementation uses approximately 5 minutes to compute the exchange, i.e., to check bounds and calculate payload and violations, for a source of size of ≈12M triples against a target of ≈6.6M triples.

4 Conclusion

We have presented a theory of RDF data validation which is geared towards protecting datasets against unwanted data, expressed in terms of what data the datasets already contain. Reservations are formalized as bounded homomorphisms of a source or contributor

into the union of the source and target graphs, in effect ensuring the merger of the two respects each component graph as specified by the homomorphism. Bound sets may be published, combined and re-used using the 'Boundz' vocabulary. Checking conformance with a bound set is computationally tractable.

The possible uses for bounds we have currently identified are automatic validation of incoming data, identifying conservative extensions of simple ontological schemas, 'write-protecting' (parts of) datasets—with different degrees of strength, and simple implementations of trust, e.g., ignoring sources that do not meet specific bounds.

Even though our presentation does not address all the requirements of a future validation standard for RDF, we believe that our work may provide valuable input to the workshop.

References

1. Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.
2. Ariel Fuxman et al. "Peer Data Exchange". In: *ACM Trans. on Database Systems* 31.4 (2006), pp. 1454–1498.
3. S. Ghilardi, C. Lutz, and F. Wolter. "Did I Damage my Ontology? A Case for Conservative Extensions in Description Logics". In: *Proc. of the 10th Int. Conference on Principles of Knowledge Representation and Reasoning (KR'06)*. 2006.
4. Aidan Hogan, Andreas Harth, and Axel Polleres. "Scalable Authoritative OWL Reasoning for the Web". In: *International Journal on Semantic Web and Information Systems* 5.2 (2009), pp. 49–90.
5. C. Lutz, D. Walter, and F. Wolter. "Conservative extensions in expressive description logics". In: *Proc. of IJCAI-07*. 2007.
6. Audun Stolpe and Martin G. Skjæveland. "Preserving Information Content in RDF Using Bounded Homomorphisms". In: *ESWC*. Ed. by Elena Simperl et al. Vol. 7295. LNCS. Springer, 2012, pp. 72–86.