Bounded RDF Data Transformations

Martin G. Skjæveland
martige@ifi.uio.no

Audun Stolpe
audus@ifi.uio.no

Department of Informatics
University of Oslo, Norway

ABSTRACT

RDF data transformations are transformations of RDF graphs to RDF graphs which preserve in different degree the data content in the source to the target. These transformation therefore give special attention to the data elements in such graphs—under the assumption that data elements reside in the subjects and objects of RDF triples, and the peculiar fact that the set of vertices and set of edges in an RDF graph are not necessarily disjoint. Bounded homomorphisms are used to define these transformations, which not only ensure that data from the source is structurally preserved in the target, but also require, in various ways, the target data to be related back to the source. The result of this paper is a theoretical toolkit of transformation characteristics with which detailed control over the transformation target may be exercised. We explore these characteristics in two different RDF graph representations, and give an algorithm for checking existence of transformations.

1. INTRODUCTION

One of the great benefits of RDF for the purposes of sharing information is its simple and uniform structure: If one has two separate RDF graphs with a consistent system of identifiers, then one can merge the two graphs simply by taking their union. This is because an RDF graph has no privileged element—*it's all triples!*—whence each RDF triple is a meaningful piece of information in its own right. Moreover, if an entity occurs as, say, a subject in two distinct triples, the two triples will merge around their common element, and the result is still a (connected) RDF graph.

Notwithstanding this, the method of taking unions will often be too primitive if one also wants a uniform representation of the data thus collected. Say for instance that one wishes to integrate information about researchers and their affiliates from various disparate sources. The domain of persondata is rich in examples of overlapping vocabularies, e.g., "Friend of a Friend" (FOAF) [4] and "Semantic Web for Research Communities" (SWRC) [20]. Suppose one wants all the information one collects to be represented in terms of the FOAF vocabulary, although some sources are marked up with SWRC. Then a simple union is not helpful. Rather one would need a way of transforming data by swapping SWRC elements for FOAF elements whilst, as far as it goes, preserving the information content of all the involved sources. A different setup of the similar problem in need of the same mechanics is the case of transforming data from other formats than RDF, say tabular data as from spreadsheets or relational databases, and possibly multiple sources, to RDF. There exists many tools which can do this translation.[1] However, such a translation is either every naive, typically translating each row in a spreadsheet into an isomorphic star-shaped RDF graph, or require manual design. In the first approach, the result is obviously a faithful representation of the original data, but the structure of the data will be very crude and is not likely to satisfy the requirements posed by a RDF vocabulary suitable for the data in question. By using a hand-crafted mapping specification and an apt tool one can shape the data into the vocabulary one wants, but now the answer to the important question of whether the translated data remains a faithful representation of the original sources is necessarily no longer apparent. Both these scenarios appear frequently in the process of consuming and publishing Linked Data, and a proper theoretical foundation for deciding such questions will help motivate transformation choices and increase the overall quality of Linked Data.

There is more than one way to characterise this problem. Although each amounts to the same formally, they tend to give rather different gestalts to the central issue. Looked at from one angle, our problem description is a data exchange problem: Given one source of data marked up in one way, one wants to migrate the data to some target repository in a way that conforms to the target's schema. Yet, it differs from the problem studied in [5] in that our setup takes the target to be fixed and possibly non-empty. Looked at from another angle, the problem concerns how to *extend* an RDF graph conservatively. More specifically, it concerns the problem of how to ensure that a transformation of source data into a target repository does not interfere with the assertive content of the source. Yet, it is unlike logic-based conservative extensions [7, 11, 15] in that the logical vocabulary is being replaced as the source is 'extended' into the target.

In this paper, we will tend to prefer the latter point of view and to speak of conservative *transformations* of a source into a target. The same problem is studied in [17, 19], but with an emphasis on SPARQL [16] construct queries as one standardised means of repurposing RDF data, and with the requirement that data elements are copied unchanged from source to target. The scope of the present paper is more general: We study the class of RDF data transformations defined from homomorphisms that substitute one predicate for another throughout a set of RDF triples and map subjects and objects injectively from source to target—under the condition that the predicate in question is not also a subject or object. We shall call these homomorphisms p-

[1] See, e.g., `http://www.w3.org/wiki/ConverterToRdf`.

maps and we measure the effects on the RDF graphs themselves. In particular, we shall show how different bounds on the p-maps reflect different ways that two RDF graphs may interlock in a reciprocal simulation that may deemed conservative in the same core sense. The aim of this exercise is to give a fuller exposition of p-maps as such, and a more comprehensive inventory of the kind of bounds that one might wish to consider. These bounds may be used to exercise detailed and differentiated control over how the vocabulary of RDF graphs is transformed into that of another: Different predicates may be restricted in different ways depending on the intended interpretation of those predicates. Our homomorphisms differ from those found in [2, 1] which essentially rename blank nodes in order to mimic the semantics of RDF as defined in [9]. Instead our definition of homomorphism resemble and is built upon standard graph homomorphisms [10]. To the best of our knowledge, our particular notion of an RDF homomorphism and the use of it is novel.

The paper is organised as follows: The next section is the prominent section of the paper and is where we set our notion of RDF graphs and homomorphisms and bounded maps of such graphs, and give the results for transformations defined from these maps. Section 3 gives a different formulation of a restricted version of p-maps and an algorithm for deciding existence of such p-maps of RDF graphs. Section 4 ends the paper with a summary and ideas for future work.

2. RDF TRIPLESETS, HOMOMORPHISMS AND BOUNDS

We first introduce a minimum of RDF [12, 9] in order to setup our formal apparatus. Let U, B and L respectively denote pairwise disjoint, fixed and infinite sets of *RDF URI references*, *blank nodes* and *literals* [12], and let $\mathcal{U} := U \cup B \cup L$ denote the union of these sets. Define the set of *RDF triples* as the set $\mathcal{T} := (U \cup B) \times U \times \mathcal{U}$. An RDF triple is commonly written as a sequence of its elements, e.g., $\vec{t} := \langle a, p, b\rangle$. An *RDF tripleset* S is a finite set of RDF triples, $S \subset \mathcal{T}$. The set of *vertices* of an RDF triple $\vec{t} := \langle a, p, b\rangle$, denoted $V(t)$, is the set $\{a, b\}$. The set of *edges* of $\vec{t}$, denoted $E(\vec{t})$, is the set $\{p\}$. We naturally extend these notions to triplesets; the set of vertices and edges of an RDF tripleset S is defined as follows: $V(S) := \cup_{\vec{t}\in S} V(\vec{t})$ and $E(S) := \cup_{\vec{t}\in S} E(\vec{t})$. We let $\mathcal{U}(S) := V(S) \cup E(S)$ denote the set of all elements occurring in S.

REMARK 1. *Even though we use standard graph terminology with the terms 'vertex' and 'edge' for the elements occurring in RDF triples, RDF graphs, as defined in [12], are not graphs in the common sense, but sets of triples. Even so, these sets are often represented and considered as graphs. The problem arises when a predicate also occurs in subject or object position, e.g., as in the (axiomatic) triple* $\langle$`rdf:type`, `rdf:type`, `rdf:Property`$\rangle$ *[9]. Figure 2 illustrates both why the graph representation is natural, and when it falls short. To highlight this subtlety and avoid any confusion, we choose to call RDF graphs* RDF triplesets*, as is done in [8].*

2.1 RDF Homomorphisms

We now introduce one of the most central constructions of this paper:

DEFINITION 1 (RDF HOMOMORPHISM). *Let S and T be RDF triplesets, then an RDF homomorphism h of S to T is a function $h : \mathcal{U}(S) \to \mathcal{U}(T)$ which induces a function $h : S \to T$ such that $h(\langle a, p, b\rangle) = \langle h(a), h(p), h(b)\rangle \in T$.*

Let h be an RDF homomorphism of S to T. We let $h(S)$ denote the RDF tripleset $h(S) := \{h(\vec{s}) \mid \vec{s} \in S\}$. It is clear that $h(S) \subseteq T$. RDF homomorphisms, as homomorphisms, reflect the structure of the source in the target. However, they are extremely undemanding with respect to the precise form of this simulation, as the following example shows:

EXAMPLE 1. *Let S be some (non-trivial) RDF tripleset, e.g., the set of triples in some FOAF file and T the singleton RDF tripleset $T := \{\langle u, u, u\rangle\}$. Then the mapping $h := \{a \mapsto u \mid a \in \mathcal{U}(S)\}$ is an RDF homomorphism of S to T.*

Needless to say, therefore, an RDF homomorphism cannot in general be said to preserve the information content of its domain—in any sense we can make of the term 'information content'. Hence, we need to look for a more restrictive subset. As a minimal requirement, we should not allow our homomorphisms to identify distinct nodes, i.e., they should be injective on vertices. This is because we adopt the operative assumption that data reside primarily in the vertices of RDF triples, and transformations must not collapse data as is done in Example 1. It is also important to note that our transformations only consider the graph-like structure of the source (and also the target, as we shall see later) and not the semantics of possible vocabularies used by the RDF triplesets. Hence, our notion of faithful or conservative transformation is purely structural and does not rely on the existence of well-defined RDF vocabularies or reasoning capabilities. It is the task of p-maps to preserve the consistency and systematicity of the relations between data elements:

DEFINITION 2 (p-MAP). *A p-map h of RDF tripleset S to RDF tripleset T is an RDF homomorphism $h : S \to T$ where $h(u) = h(v)$ implies $u = v$, for all $u, v \in V(S)$.*

For the remainder of this subsection we record some simple descriptive facts about p-maps which are immediate from Definition 1 and Definition 2. Requiring that h is injective on the vertices in the source ensures that the source data in is embedded into the target, or, stated differently, the target data (possibly) extends the source data:

PROPOSITION 1. *If h is a p-map $h : S \to T$, then $V(h(S)) \subseteq V(T)$.*

The set of vertices and set of edges of an RDF tripleset need not be disjoint. This is an idiosyncratic and important feature of RDF which, as we shall see, will have ramifications throughout this paper. For now we record that edges which also appear as vertices in the source must abide by the condition placed on vertices. This again affects triples containing such edges forcing them to be mapped injectively from source to target:

PROPOSITION 2. *Let S be an RDF tripleset and h any p-map of S, then $h(\vec{t_1}) = h(\vec{t_2})$ implies $\vec{t_1} = \vec{t_2}$ for RDF triples $\vec{t_1}, \vec{t_2}$ such that $E(\vec{t_1}) \subseteq V(S)$ and $E(\vec{t_2}) \subseteq V(S)$.*

A natural strengthening of the injective mapping of the source vertices is to require that they are mapped by the identity function. Mapping data with the identity function may not seem to be much of a transformation, however, in many cases this is the correct choice, at least for source vertices occupied by instance data and not vocabulary elements:

EXAMPLE 2. *Consider the following source RDF tripleset and assume it is to be transformed into using the FOAF vocabulary.*[2]

```
dbp:Bernstein  rdf:type         swrc:Person ;
               swrc:supervisor  dbp:Ullman .
```

The p*-map*

$$\begin{aligned} h = \{&\texttt{dbp:Bernstein} \mapsto \texttt{dbp:Bernstein},\\ &\texttt{dbp:Ullman} \mapsto \texttt{dbp:Ullman},\\ &\texttt{rdf:type} \mapsto \texttt{rdf:type},\\ &\texttt{swrc:Person} \mapsto \texttt{foaf:Person},\\ &\texttt{swrc:supervisor} \mapsto \texttt{foaf:knows}\} \end{aligned}$$

represents a probable transformation of this dataset, resulting in the following target RDF tripleset:

```
dbp:Bernstein  rdf:type    foaf:Person ;
               foaf:knows  dbp:Ullman .
```

Note that what can be considered the data elements in this dataset, dbp:Bernstein *and* dbp:Ullman*, commonly also called* individuals *or* instances*, are mapped with the identity function. In the examples that follow we will map elements in the source not listed in the specification of the transformation with the identity function. These elements will always be what we coin 'data elements'.*

Under the assumption that p-maps require the identity mapping of source vertices they exhibit some especially strong features, which follow easily from the preceding propositions:

PROPOSITION 3. *If h is a* p*-map $h : S \to T$ where $h(u) = u$ for all $u \in V(S)$, then $V(S) \subseteq V(T)$.*

PROPOSITION 4. *Let S be an RDF tripleset and h any* p*-map of S where $h(u) = u$ for all $u \in V(S)$, then $h(\vec{t}) = \vec{t}$ for every RDF triple $\vec{t}$ such that $E(\vec{t}) \subseteq V(S)$.*

As per Definition 2, p-maps are still too lenient a notion of simulation, as they do not prevent interference with information in the source (recall that we are viewing our problem as a problem of conservatively extending the source into the target). Such interference arises in the case where the target already contains data elements equal to those coming from the source, and where the source is rewritten into the vocabulary that relates those elements. In such cases the transformation may have unexpected side effects:

EXAMPLE 3. *Put $S := \{\langle a, p, b\rangle\}$, $T_1 := \{\langle b, q, a\rangle\}$ and $T_2 := \{\langle a, q, b\rangle\}$. Let h be the identity on a and b and put $h(p) = q$. Then using h to rewrite S into T_2 produces T_2 itself, which is unproblematic if one already deems q a suitable replacement for p. However, rewriting S into T_1 yields the tripleset $\{\langle a, q, b\rangle, \langle b, q, a\rangle\}$ which is not necessarily consistent (in an intuitive sense) with S. For instance, put $p =$* ex:doctoralAdvisor *and $q =$* swrc:supervisor*. Then S says about a that his or her doctoral advisor is b, whilst the extension of S into T_1 modulo h says that a and b are each others' doctoral advisors.*

2.2 Bounds

Example 3 shows that a transformation of a source into a target needs to be sensitive to the information already contained in the target in order to exclude those targets that are intuitively incompatible with the source. What is needed is a way to reflect the structure of the target back into the source in such a way that two RDF triplesets so related may be said to interlock in a reciprocal simulation that is conservative in some desired sense. This is where our announced bounds enter the picture:

DEFINITION 3 (BOUNDED p-MAP). *A* p*-map $h : S \to T$ is* bounded*, and called a* p1-*,* p2- *or* p3-map*, respectively, if it satisfies one of the following conditions. For all $a, p, b \in \mathcal{U}$:*

$$\langle a, h(p), b\rangle := \vec{t} \in T \quad \Rightarrow \quad \exists \vec{s} \in S \,.\, h(\vec{s}) = \vec{t} \qquad (\mathsf{p1})$$

$$\begin{aligned} &\langle a, h(p), h(b)\rangle := \vec{t} \in T \quad \textit{or}\\ &\langle h(a), h(p), b\rangle := \vec{t} \in T \quad \Rightarrow \quad \exists \vec{s} \in S \,.\, h(\vec{s}) = \vec{t} \qquad (\mathsf{p2}) \end{aligned}$$

$$\langle h(a), h(p), h(b)\rangle := \vec{t} \in T \quad \Rightarrow \quad \exists \vec{s} \in S \,.\, h(\vec{s}) = \vec{t} \qquad (\mathsf{p3})$$

The next theorem records the effect the different bounds on p-maps have on the target of the transformation:

THEOREM 1. *Let S and T be RDF triplesets, $h : S \to T$ a* p*-map, and $a, p, b \in \mathcal{U}$. Then the following hold:*

1. *h is a* p1*-map iff there is no $\langle a, h(p), b\rangle \in T \setminus h(S)$.*
2. *h is a* p2*-map iff for all $\vec{t} := \langle a, h(p), b\rangle \in T \setminus h(S)$ we have $V(\vec{t}) \cap V(h(S)) = \emptyset$.*
3. *h is a* p3*-map iff for all $\vec{t} := \langle a, h(p), b\rangle \in T \setminus h(S)$ we have $V(\vec{t}) \nsubseteq V(h(S))$.*
4. *h is a bounded* p*-map iff $\langle h(a), h(p), h(b)\rangle \in T$ implies $\langle a, p, b\rangle \in S$.*

PROOF. We prove only the claim for p1-maps, the proofs for the other bounds are similar.

$\Rightarrow$) Assume h is a p1-map and let $\vec{t} := \langle a, h(p), b\rangle \in T$ for some $a, p, b \in \mathcal{U}$. We need to show that $\vec{t} \in h(S)$. By (p1) there is a triple $\vec{s} \in S$ such that $h(\vec{s}) = \vec{t}$. Set $c := h(a)$ and $d := h(b)$ for $c, d \in V(S)$ giving $\vec{s} = \langle c, p, d\rangle$, since h is a p-map which must preserve the structure of triples and is injective on vertices. It follows that $h(\vec{s}) = \vec{t} \in h(S)$.

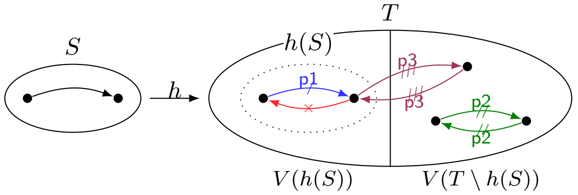
$\Leftarrow$) Assume there is no triple $\langle a, h(p), b\rangle \in T \setminus h(S)$ and let $\vec{t} := \langle a, h(p), b\rangle \in T$ for some $a, p, b \in \mathcal{U}$. Then, by assumption, $\vec{t} \in h(S)$, so, since h is a p-map, there must be a triple $\vec{s} \in S$ such that $h(\vec{s}) = \vec{t}$. □

COROLLARY 1. *Condition (*p1*) is strictly stronger than (*p2*), and (*p2*) is strictly stronger than (*p3*).*

[2] In all code examples we assume prefixes are correctly set:

swrc:	http://swrc.ontoware.org/ontology#,
dbp:	http://dbpedia.org/resource/,
foaf:	http://xmlns.com/foaf/0.1/,
vcard:	http://www.w3.org/2006/vcard/ns#, and
ex:	http://example.net/.

For brevity, first names have been removed from identifiers, e.g., the identifier dbp:Ullman should be dbp:Jeffery_Ullman.



The figure illustrates the transformation of a singleton tripleset S into the target T with the map h. Triples are illustrated by letting arrows represent the edge between two vertices, which appear as dots in the diagram. The image of S under h is indicated by the dotted ellipse. The set of vertices in the target is partitioned in two: $V(h(S))$ and $V(T \setminus h(S))$. All arrows in the target represent edges in the image of h. The labelled arrows show triples which are typically permissible under the bound indicated by its label: A p1-map (−/→) restricts the set of triples with edges in the image under h to only those triples (which are already) in the image of S under h. A p2-map (−//→) also allows triples with edges in the image under h if none of the vertices are in $V(h(S))$, i.e., a p2-map may not relate a vertex in $V(h(S))$ with a vertex $V(T \setminus h(S))$ (or vice versa). A p3-map (−///→) additionally allows triples with edges in the image under h if one of the vertices is in $V(h(S))$. No bounded p-map allows triples with edges in the image under h to relate vertices in $V(h(S))$ if they are not related by the mapped edge in the source (−×→). Bounded maps only control target triples with edges in the image under h, therefore are only such triples illustrated.

Figure 1: Bound characteristics exemplified.

Proof. Follows easily from Theorem 1. □

Theorem 1 shows that bounded p-maps restrict, with different degree of strength, the occurrences of triples in the target using predicates to which elements from the source are mapped; see Figure 1 for an exemplification of the theorem. This does not mean that the target may not contain triples not coming from source; the target may contain more information in the form of triples, as long as these triples do not have source edges that map to them. Indeed, a p1-map need not even be injective:

Example 4. *Assume the RDF triplesets $S := \{\langle a,p,b\rangle, \langle a,q,b\rangle\}$ and $T := \{\langle a,r,b\rangle, \langle c,s,d\rangle\}$. Then $\{p \mapsto r, q \mapsto r\}$ is a* p1*-map of S to T, but it is neither onto nor injective.*

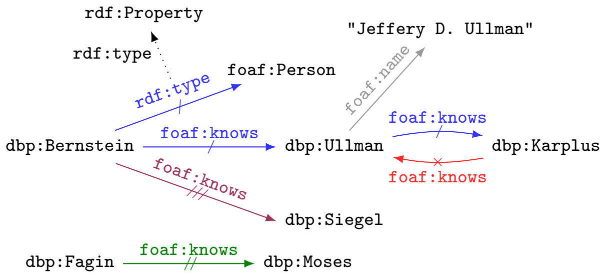
As the example illustrates, edges from the source tripleset may be identified in the target if they relate the exact same set of pairs in the source. The identification of such edges may be thought of as a kind of *limited generalisation*.

The next example illustrates the use of bounded p-maps and implications of Theorem 1 on “real” data.

Example 5. *Let S be the following RDF tripleset*

```
dbp:Bernstein  rdf:type          swrc:Person ;
               swrc:supervisor   dbp:Ullman .
dbp:Ullman     swrc:supervisor   dbp:Karplus .
```

and let h be the p*-map $h = \{$*`swrc:supervisor` $\mapsto$ `foaf:knows`, `rdf:type` $\mapsto$ `rdf:type`, `swrc:Person` $\mapsto$ `foaf:Person`$\}$. *We now give permissible target RDF tripleset for transformations under different—or no—bounds on h. To easily create different RDF triplesets, let T_x denote the RDF tripleset obtained by collecting the lines from the code listing below as indicated by x, where x is an interval of line numbers, e.g., $T_{1\text{–}3}$ is the RDF tripleset consisting of the triple in lines 1, 2, 3. The tripleset below is illustrated in Figure 2.*



The solid arrows span the graph which is the common way of representing the RDF tripleset given in Example 5. The dotted arrow represents the RDF axiomatic triple ⟨**rdf:type**, **rdf:type**, **rdf:Property**⟩ [9]. Given the transformation setting from this example, blue arrows (−/→) indicate triples which are required under a p-map of the source to the target, and are the only ones permissible using the predicates **rdf:type** and **foaf:knows** under a p1-map. The green arrow (−//→) is permissible for a p2-map, while the purple (−///→) is allowed under a p3-map. The red arrow (−×→) is not permissible under any bounds on the map. The grey arrow (—→) is not affected by the specified map.

Figure 2: Visualisation of an RDF tripleset, and bounded p-maps.

```
1  dbp:Bernstein  rdf:type    foaf:Person ;
2                 foaf:knows  dbp:Ullman .
3  dbp:Ullman     foaf:knows  dbp:Karplus .
4
5  dbp:Fagin      foaf:knows  dbp:Moses .         # not p1
6  dbp:Bernstein  foaf:knows  dbp:Siegel .        # not p2
7  dbp:Karplus    foaf:knows  dbp:Bernstein .     # not p3
8  dbp:Ullman     foaf:name   "Jeffery D. Ullman" .
```

Recall Corollary 1 and the logical implications between the bounded and unbound p*-maps: all* p1*-maps are* p2*-maps, all* p2*-maps are* p3*-maps, and all* p3*-maps are* p*-maps. Any RDF tripleset which is a superset of $T_{1\text{–}3}$ is a permissible tripleset target under h with no bounds, specifically, the tripleset $T_{1\text{–}8}$ is a valid* p*-map target. An RDF tripleset not containing the triples in $T_{1\text{–}3}$ will not satisfy the homomorphism condition of* p*-maps. The tripleset $T_{1\text{–}5}$ is not a valid target if h is set to be a* p1*-map. Such maps do not allow new data elements to be introduced by edges in the image of h—new data elements meaning elements not mapped to from the source, e.g., both* `dbp:Fagin` *and* `dbp:Moses` *are “new” as they occur in the target not as a result of the transformation of the source. The same RDF tripleset is however permissible under bound (*p2*). This condition allows introduction of such new elements, as long as triples which contain new elements do so in both vertices, i.e., mixing new and old data elements is not allowed. Since* `dbp:Fagin` *and* `dbp:Moses` *do not occur in the source, the tripleset $T_{1\text{–}5}$ is a valid target, while $T_{1\text{–}6}$ is not, given that* `dbp:Bernstein` *does occur in the source and* `dbp:Siegel` *does not. However, this tripleset does not break condition (*p3*). Maps of type* p3 *allow mixing new and old elements as long as the old element occurs in the same position in the target as in the source, e.g.,* *since* $\langle\texttt{dbp:Bernstein}, \texttt{foaf:knows}, \texttt{dbp:Ullman}\rangle \in h(S)$*, then* $\langle\texttt{dbp:Bernstein}, \texttt{foaf:knows}, \texttt{dbp:Siegel}\rangle$ *is an acceptable target triple under bound* (p3)*. The triple in line 7 is not permissible in a target under any bounds on h. No bounded* p*-map allows old data elements to be related in new ways by edges originating from the source. On the other hand, the triple in line 8 is accepted by all bounds on the transformation described by h. The reason being that* `foaf:name` *is not in the range of h and* p*-maps may only control triples containing edges in its range.*

The important feature of bounded p-map is that they forbid new relationships between data from the source using target representatives of source edges, and that data which does not originate from the source may use these representatives only according to the bound on the map: p1-maps are suited for those parts of a dataset to which one would wish to remain absolutely faithful, typically the domain-specific information that is collected and managed by the issuer of the dataset, i.e., data which does not originate from the source may not use p1-mapped edges at all. A p2-map could be used when domain-specific knowledge is to be merged from two different sources whilst keeping the information from each of the sources unchanged. It is more forgiving than a p1-map since it allows a relation to grow as long as every added pair relates new elements only. Maps of type p3 are typically applied to vocabulary elements considered a part of the general-purpose vocabulary or to relations one wants to extend in the transformation. For instance, applied to `rdf:type`, it allows types to be added to source elements—given that those types are not already represented in the source, since, as illustrated by Example 3, this can disturb the representation of the source in the target.

While it is fairly immediate that bounded p-maps can be an instrument for data integration and data exchange of RDF data, observe that bounded p-maps may also play a role in *data fusion*, which is defined as "the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation" [3]. Although bounded p-maps offer no mechanics for how to decide what data element to choose for a specific property when different values are available from the different sources and exactly one is needed, it does allow one to specify for which properties, i.e., edges, such a choice is required (by requesting a p1- or p2-map) or not (by using a p3-map—or even an unbounded p-map).

The next example shows how we can exercise differentiated control over the transformation target by applying different bounds to different edges in source:

Example 6. *Assume the RDF tripleset S contains the following three triples:*

```
dbp:Fagin  swrc:firstName      "Ronald" .
dbp:Fagin  swrc:lastName       "Fagin" .
dbp:Fagin  swrc:cooperateWith  dbp:Moses .
```

Let h be a p*-map of S:*

$$\begin{aligned} h = \{&\texttt{swrc:firstName} \mapsto \texttt{foaf:firstName},\\ &\texttt{swrc:lastName} \mapsto \texttt{foaf:surname},\\ &\texttt{swrc:cooperateWith} \mapsto \texttt{foaf:knows}\} \end{aligned}$$

It makes good sense to translate the mapping of `swrc:cooperateWith` ↦ `foaf:knows` *under* (p3)*, allowing Fagin to add more friends than just his companion, while binding the translation of his names under* (p1) *or* (p2)*, depending on whether the data is to be merged with other data or not, to ensure that Mr. Fagin does not gain more names under the transfer.*

So far we have established that p-maps may be used as a formal instrument for structured and non-distortive transformations of RDF data from source to target under application of a single mapping. Next, we show that composing bounded p-maps yields bounded p-maps, meaning p-maps can be used to ensure a safe transformation the RDF data also through multiple mappings:

Theorem 2. *Let S, T and R be RDF triplesets and* $h_1 : S \to T$ *and* $h_2 : T \to R$ *be bounded* p*-maps. Then* $h = h_2 \circ h_1$ *is a bounded* p*-map of S to R, satisfying the weakest bound of those on* h_1 *and* h_2*.*

Proof. We need to check that h is a homomorphism, that it satisfies the p-map condition, and lastly, that it satisfies the weakest bound of h_1 and h_2. It is well-known that the composition of homomorphisms is a homomorphism, see, e.g., [10]. It is also clear that h must be a p-map, since composing injective functions yields an injective function. By Corollary 1 the question of whether h satisfies the weakest bound can be simplified to the case where h_1 and h_2 are restricted by the same bound. So assume that h_1 and h_2 are p1-maps and $\vec{r} =: \langle a_R, h_2(h_1(p)), b_R\rangle \in R$; we need to show that there is a $\vec{s} \in S$ such that $h(\vec{s}) = \vec{r}$. Since h_2 is a p1-map, there must be a triple $\vec{t} \in T$ where $h_2(\vec{t}) = \vec{r}$, and given that h_2 is a homomorphism $\vec{t}$ be of the form $\langle a_T, h_1(p), b_T\rangle$, where $a_T, b_T \in \mathcal{U}(T)$. Continue using the same argument with $h_1 : S \to T$ to arrive at that there is a $\vec{s} = \langle a_S, p, b_S\rangle \in S$ such that $h_1(\vec{s}) = \vec{t}$, and $h(\vec{s}) = \vec{r}$ as desired. The cases when the bounds are (p2) or (p3) are similar. □

Example 7. *Assume the following tripleset S:*

```
dbp:Ullman     swrc:supervisor  dbp:Karplus .
```

is transformed into the tripleset T below with the p2*-map* $h_1 := \{\texttt{swrc:supervisor} \mapsto \texttt{swrc:cooperateWith}\}$.

```
dbp:Ullman     swrc:cooperateWith  dbp:Karplus .

dbp:Fagin      swrc:cooperateWith  dbp:Moses .
dbp:Moses      swrc:cooperateWith  dbp:Fagin .
```

This tripleset is in turn transformed using the p3*-map* $h_2 := \{\texttt{swrc:cooperateWith} \mapsto \texttt{foaf:knows}\}$ *to the target R:*

```
1  dbp:Ullman     foaf:knows  dbp:Karplus .
2  dbp:Fagin      foaf:knows  dbp:Moses .
3  dbp:Moses      foaf:knows  dbp:Fagin .
4
5  dbp:Ullman     foaf:knows  dbp:Vardi .
6  dbp:Fagin      foaf:knows  dbp:Vardi .
7  dbp:Vardi      foaf:knows  dbp:Moses .
```

These triplesets and maps are illustrated in Figure 3. By Theorem 2 the composition of the two maps $h_2 \circ h_1 = \{\texttt{swrc:supervisor} \mapsto \texttt{foaf:knows}\}$ *is a* p3*-map of S to R. Note that if R did not contain the triple in line 5:* $\langle\texttt{dbp:Ullman}, \texttt{foaf:knows}, \texttt{dbp:Vardi}\rangle$*, then the composed map would be a* p2*-map even though* h_2 *still would be a* p3*-map. The fact that the composition would then be a* p2*-map is easily seen in Figure 3, where the triples in* $R \setminus h_2(h_1(S))$ *(and excluding the given triple, which is marked by* $*$ *in the figure) would not be connected to the triples coming from S.*

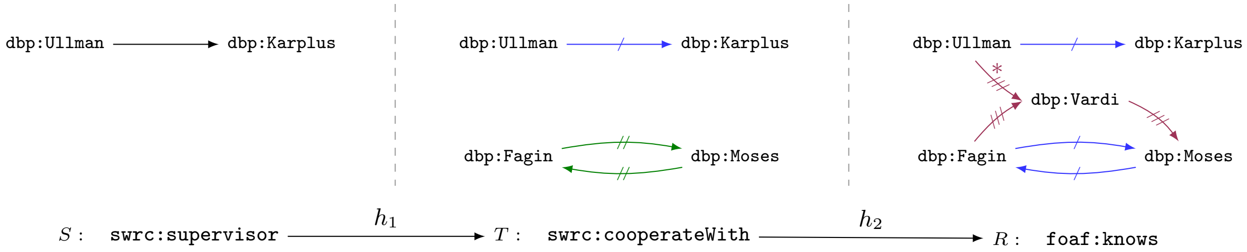


Illustration of the three triplesets S, T and R, and p-maps $h_1 : S \to T = \{\texttt{swrc:supervisor} \mapsto \texttt{swrc:cooperateWith}\}$ and $h_2 : T \to R = \{\texttt{swrc:cooperateWith} \mapsto \texttt{foaf:knows}\}$ used in Example 7. The triplesets are divided by a dashed line, and the URI below each graph indicate the label which is used for all the edges in the graph. A blue ($\not\rightarrow$), green ($/\!\!/\!\!\rightarrow$) and purple ($/\!\!/\!\!/\!\!\rightarrow$) edge specify that the triple satisfies respectively the bounds (p1), (p2) and (p3) when setting the tripleset immediate to the left as source for the map. The edge marked by $*$ in the tripleset R is addressed by a comment in Example 7.

Figure 3: Composing p-maps.

2.3 Adding Strength to p-maps

As recorded by Example 5, bounded p-maps control only edges in its range. An obvious strengthening of p-maps is therefore to require that the range of the map includes all the edges of the target. This is easily done by slightly extending the bounds of Definition 3 and proving the results equivalent to those of Theorem 1 for the new bounds:

Definition 4 (p$^+$-map). *A p$^+$-map $h : S \to T$ is such that $E(T) \subseteq \mathsf{range}(h)$, and p1$^+$-, p2$^+$- and p3$^+$-maps are p$^+$-maps satisfying respectively bounds (p1), (p2) and (p3). We call these bounds (p1$^+$), (p2$^+$) and (p3$^+$), respectively.*

Theorem 3. *Let S and T be RDF triplesets, $h : S \to T$ a p$^+$-map, and $a, p, b \in \mathcal{U}$. Then the following hold:*

1. *h is a p1$^+$-map iff there is no RDF triple $\langle a, p, b\rangle \in T \setminus h(S)$ (i.e., $h(S) = T$).*
2. *h is a p2$^+$-map iff there is no RDF triple $\langle a, p, b\rangle \in T \setminus h(S)$ where $\{a, b\} \cap V(h(S)) \neq \emptyset$.*
3. *h is a p3$^+$-map iff there is no RDF triple $\langle h(a), p, h(b)\rangle \in T \setminus h(S)$.*

Proof. We show only the claim for (p3$^+$), the proofs for (p1$^+$) and (p2$^+$) is similar. Assume h is a p3$^+$-map, then, by (p3) and Definition 4, for all triples $\vec{t} := \langle h(a), p, h(b)\rangle \in T$ where $a, b \in V(h(S))$ we have $h(\vec{s}) = \vec{t}$ for some $\vec{s} \in S$. Thus there can be no triple in $T \setminus h(S)$ where both vertices do not occur in $h(S)$. For the other direction suppose there is no triple $\langle h(a), p, h(b)\rangle \in T \setminus h(S)$ where $a, b \in V(h(S))$. Then for every $\vec{t} := \langle h(a), q, h(b)\rangle \in T$ there is a $p \in E(S)$ such that $h(\langle a, p, b\rangle) = \vec{t} \in h(S)$ for every $a, b \in V(S)$, which means that h must be a p3$^+$-map. □

Example 8. *Let S, T_x and h be as specified in Example 5. Since we have $|E(S)| < |E(T_{1\text{–}8})|$, there is no p$^+$-map of S to $T_{1\text{–}8}$. However, the transformations $S \to T_{1\text{–}5}$, $S \to T_{1\text{–}6}$, $S \to T_{1\text{–}7}$ are bounded by (p1$^+$), (p2$^+$) and (p3$^+$), respectively.*

Bounds on p$^+$-maps relate to bounds on p-maps in the following way:

Lemma 1. *The bound (pn$^+$) is strictly stronger than the bound(pn), for $\mathsf{n} = 1, 2, 3$.*

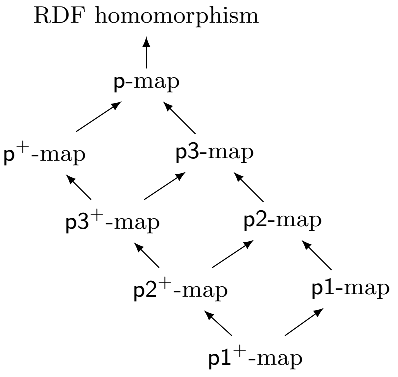


Figure 4: Partial order over inclusion of the set of classes of maps introduced in this section.

Proof. Follows by straight-forward comparison of Theorem 1 and Theorem 3. □

Now, by transitivity of Corollary 1 and Lemma 1, these results may be compiled to the lattice in Figure 4 showing the inclusion relationships between the different class of maps introduced in this section. Note, however, that p-maps and p$^+$-maps may not be applied in conjunction to produce a stratified map of a source to target transformation in the same sense as is for p-maps shown in Example 6. The reason being that p$^+$-maps require a "global" restriction on the map, that is, that the map range over all the edges in the target. That is, if some edges in the source are p-mapped and others p$^+$-mapped, it would mean that all p-mapped edges are mapped with a p$^+$-map (which of course may be perfectly fine).

We end this section by describing the strength of p1$^+$-maps, the strongest bounded map introduced, in terms of equivalence of maps:

Definition 5 (Map Equivalence). *Two RDF triplesets S and T are* χ-map equivalent *if there is there is a χ-map $S \to T$ and a χ-map $T \to S$, for $\chi \in \{$p, p1, p2, p3, p$^+$, p1$^+$, p2$^+$, p3$^+\}$.*

Theorem 4. *If $h : S \to T$ is a p1$^+$-map, then S and T are p1-map equivalent.*

PROOF. By Lemma 1 h is a $\mathsf{p1}$-map, so we need only to show that there is a $\mathsf{p1}$-map $g : T \to S$. We construct g as follows: Set $g(u) = h^-(u)$ for all $u \in V(T)$; this is well-defined as h is injective on the vertices in S by Definition 2. For each $p \in E(T) \setminus V(T)$ choose one $q \in E(S)$ such that $h(q) = p$ and set $g(p) = q$; this is well-defined as h is surjective on the edges in T. By Theorem 3 it is clear that g is defined for all $u \in \mathcal{U}(T)$. By construction of g and Theorem 1 it is clear that g is a $\mathsf{p1}$-map. □

For a $\mathsf{p2}^+$-map we may have the situation where $V(h(S)) \subset V(T)$, which means, by Proposition 1, that there can be no p-map $T \to S$ as not all vertices are carried over from T to S. In the case that the transformation is a $\mathsf{p1}$-map, we may have that $|E(S)| < E(T)|$, by which there may be no p-map of the source to the target. Here is an example:

EXAMPLE 9. *Let $S := \{\langle a, p, b\rangle\}$, $T_1 := \{\langle a, q, b\rangle, \langle c, q, d\rangle\}$ and $T_2 := \{\langle a, q, b\rangle, \langle c, r, d\rangle\}$ be RDF triplesets and $h := \{p \mapsto q\}$ a map. Then, h is a $\mathsf{p2}^+$-map of S to T_1, but there is no p-map of T_1 to S. Similarly, h is a $\mathsf{p1}$-map of S to T_2, but there is no p-map of T_2 to S.*

Theorem 4, Figure 4 and the previous example imply that $\mathsf{p1}^+$-maps is the only class of maps which enforce p-map equivalent source and target triplesets. A subtle point to make of this observation is that $\mathsf{p1}^+$-maps are the only class of maps introduced in this paper which force homomorphically equivalence between source and target, yet this does not mean that the more liberal bounds are not useful for the purpose of conservative transformations. On the contrary, the weaker bounds than $\mathsf{p1}^+$ seem a to be a better fit especially in the setting of RDF and Linked Data, where data is frequently merged from disparate and heterogeneous sources and will often require extra vertices and edges in the target tripleset in order to "glue" the sources into one coherent target.

3. BINARY FACTORS, INCLUSION MAPS, BOUNDS AND COMPUTATION

The purpose of the present section is to give a different representation of a restricted version of bounded p-maps, those which map vertices identically, that is more amenable to computation. Thus, in the following we adopt a stronger notion of p-maps and require that they always map vertices identically from the source to target, and call these maps simply *restricted* p*-maps*. The reason why we focus on only these restricted p-maps is that, as motivated earlier in the text and through examples, it is in many cases natural to translate instance data identically. Also, this restriction makes the problem of checking the existence of a p-map of one tripleset to another polynomial in the size of the source and target triplesets, a problem which in its general form, i.e., GRAPH HOMOMORPHISM, is NP-complete [6]. The following lemma resets the bounds to a more elegant formulation under the new restriction.

LEMMA 2. *Let h be a p-map $h : S \to T$ where $h(u) = u$ for all $u \in \mathcal{U}(S)$. Then the following bounds are equivalent to those of Definition 3. For all $a, p, b \in \mathcal{U}$:*

$$\langle a, h(p), b\rangle \in T \quad \Rightarrow \quad \langle a, p, b\rangle \in S \qquad (\mathsf{p1})$$

$$\langle a, h(p), h(b)\rangle \in T \text{ or } \langle h(a), h(p), b\rangle \in T \quad \Rightarrow \quad \langle a, p, b\rangle \in S \qquad (\mathsf{p2})$$

$$\langle h(a), h(p), h(b)\rangle \in T \quad \Rightarrow \quad \langle a, p, b\rangle \in S \qquad (\mathsf{p3})$$

PROOF. Follows from Definition 3 and Definition 2. □

We shall reduce the problem of checking the existence of a restricted p-map to the problem of checking certain set-inclusions. The central concept in this endeavour is that of a *binary factor* which is essentially an equivalence class of pairs induced by a common property which relates each pair in the class. Binary factors allow us to reason about a restricted p-map vicariously, so to speak, in terms of the data in the RDF triplesets in question: By checking the set-theoretic relations between the binary factors induced by restricted p-map associated edges we check the relationship between the edges themselves. Of course, we shall have to reformulate the bounds accordingly, and we do so by translating them into constraints on inclusion maps, i.e., into constraints on maps that take one set to another in which it is included. Although, binary factors prove a natural simplified representation of RDF triplesets and therefore interesting in it self, the main justification for this overall increase in our notational apparatus will come in Subsection 3.1, where we give an algorithm for computing restricted p-maps that it is straightforward to formulate, and easy to see is polynomial once we have binary factors at our disposal. Binary factors and inclusion maps are defined immediately below:

DEFINITION 6 (BINARY FACTOR). *Let S be an RDF tripleset. Then $[\cdot]_S : E(S) \to \mathcal{P}(V(S) \times V(S))$ is a function where $[p]_S = \{\langle a, b\rangle \mid \langle a, p, b\rangle \in S\}$. We call $[p]_S$ the* binary factor *of p in S, and denote with $[S]$ the set of binary factors in S: $[S] = \{[p]_S \mid p \in E(S)\}$.*

DEFINITION 7 (INCLUSION MAP). *Let X and Y be families of sets. A function f is an* inclusion map *from X to Y iff $f(x) \in Y$ and $x \subseteq f(x)$ for every $x \in X$.*

On the face of it, inclusion maps of binary factor sets capture the essence of restricted p-maps of RDF triplesets:

EXAMPLE 10. *Consider the restricted p-map in Example 3. The binary factors of S and T_2 from that example are $[S] = \{[p]_S\} = \{\{\langle a, b\rangle\}\}$, $[T_2] = \{[q]_{T_2}\} = \{\{\langle a, b\rangle\}\}$. There is an inclusion map $f : [S] \to [T_2]$ where $f([p]) = [q]_{T_2}$. This map corresponds to the p-map $h : S \to T_2$ in Example 3.*

However, not all inclusion maps of binary factors sets induce restricted p-maps. The case to consider is an RDF tripleset where an edge also plays the role of a vertex:

EXAMPLE 11. *Put $S := \{\langle a, a, b\rangle\}$ and $T := \{\langle a, q, b\rangle\}$, for distinct p and q. There is an inclusion map $f : [S] \to [T]$, namely the map where $f([a]_S) = [q]_T$, but $h := \{a \mapsto q\}$ is no p-map of S to T, since h must be a function on $\mathcal{U}(S)$ (and cannot send the element a to both a and q).*

In order to rule out such anomalies, we need to isolate the class of inclusion maps that preserve the property expressed by Proposition 4:

Lemma 3. *Let $f : [S] \to [T]$ be an inclusion map for RDF triplesets S and T. If there exists a function h_f induced from f by setting*

- *$h_f(p) = q$ whenever $f([p]_S) = [q]_T$ for all $[p]_S \in [S]$, and*
- *$h_f(u) = u$ for all $u \in V(S)$,*

then h_f is a restricted p*-map of S to T.*

Proof. Assume the preconditions of the lemma hold and that h_f does exist. To show the homomorphism condition holds, suppose $\langle a, p, b\rangle \in S$ and $f([p]_S) = [q]_T$, then $h_f(p) = q$. Since $\langle a, b\rangle \in [p]_S$ it follows, since f is an inclusion map, that $\langle a, b\rangle \in [q]_T$, whence $\langle a, q, b\rangle \in T$, so we have $\langle h_f(a), h_f(p), h_f(b)\rangle \in T$ as desired. Since $h_f(u) = u$ for all $u \in v(S)$, then h_f respects the identity on vertices in S, and by Definition 6, $[p]_S \in [S]$ for all edges p in S, so h_f is defined for all $u \in \mathcal{U}(S)$. □

We single out this class of inclusions maps with a separate definition, for easy reference:

Definition 8 (b-map). *Let S and T be RDF triplesets. A* b-map *is an inclusion map $f : [S] \to [T]$ if the following holds: for all $[p]_S \in [S]$, if $f([p]_S) = [q]_T$ and $p \in V(S)$, then $p = q$.*

The next result confirms the adequacy of this condition:

Lemma 4. *If $f : [S] \to [T]$ is a* b*-map, then there is a restricted* p*-map h_f of S to T induced by f defined in Lemma 3.*

Proof. Suppose $f : [S] \to [T]$ is a b-map. Construct a function $h_f : \mathcal{U}(S) \to \mathcal{U}(T)$ as follows: Put $\langle u, u\rangle \in h_f$ for each $u \in V(S)$, and for every $p \in E(S)$ choose a $q \in E(T)$ such that $f([p]_S) = [q]_T$, and put $\langle p, q\rangle \in h_f$; since f is an inclusion map, such $[p]_S$'s and $[q]_T$'s clearly exist. By Lemma 3 it suffices to show that h_f indeed is a function. Aiming for a contradiction, let $\langle p_1, q_1\rangle, \langle p_2, q_2\rangle \in h_f$ and suppose that $p_1 = p_2$ whilst $q_1 \neq q_2$. There are two cases to consider:

1. $p_1 \in V(S)$: Then, h_f being the identity on vertices in S, we have $p_1 = q_1$, so $p_1 = p_2 = q_1$. Since $q_1 \neq q_2$, by assumption, it follows that $p_2 \neq q_2$, so p_2 must be an edge in S. By construction of h_f, $f([p_2]_S) = [q_2]_T$. Given that $p_2 = p_1 \in V(S)$, the b-map condition applies, so $p_2 = q_2$; a contradiction.

2. $p_1 \in E(S)$: Then, since $p_1 = p_2$, by assumption, it follows that $f([p_1]_S) = [q_1]_T$ and $f([p_2]_S) = [q_2]_T$. Moreover, since $p_1 = p_2$ we have $[p_1]_S = [p_2]_S$, whence $[q_1]_T = [q_2]_T$ by the construction of h_f, which contradicts the assumption that $q_1 \neq q_2$.

This concludes the proof. □

Indeed, there is a one-to-one correspondence between b-maps on binary factors and restricted p-maps:

Lemma 5. *If there is a restricted* p*-map h of S to T, then there exists a* b*-map f_h of $[S]$ to $[T]$, induced by h by setting $f([p]_S) = [h(p)]_T$ for all $p \in E(S)$.*

Proof. Suppose $h : S \to T$ is a restricted p-map. It follows immediately from the p-map condition that the map defined as $f([p]_S) = [h(p)]_T$ for all $p \in E(S)$ is an inclusion map of $[S]$ to $[T]$. It remains to show that f is a b-map. Suppose $f([p]_S) = [q]_T$ and $p \in V(S)$. From the former we have $h(p) = q$, from the latter we have $h(p) = p$. Therefore $p = q$ as desired. □

Theorem 5. *There is a restricted* p*-map of S to T iff there exists a* b*-map of $[S]$ to $[T]$.*

Proof. Follows by Lemma 4 and Lemma 5. □

Turning to the bounds on restricted p-maps, they are easily expressed in terms of binary factors. Indeed, a very slight notational alteration will do:

Theorem 6. *The bounds (*p1*), (*p2*), (*p3*), (*p1$^+$*), (*p2$^+$*) and (*p3$^+$*) on restricted* p*-maps from Lemma 2 and Definition 4 correspond to the respective conditions (*b1*), (*b2*), (*b3*), (*b1$^+$*), (*b2$^+$*) and (*b3$^+$*) on* b*-maps $f : [S] \to [T]$ listed below:*

$$\forall [p]_S \in [S]. f([p]_S) = [p]_S \tag{b1}$$

$$\forall [p]_S \in [S]. \langle a, b\rangle \in f([p]_S) \setminus [p]_S \Rightarrow \{a, b\} \cap V(S) \neq \emptyset \tag{b2}$$

$$\forall [p]_S \in [S]. \langle a, b\rangle \in f([p]_S) \setminus [p]_S \Rightarrow \{a, b\} \not\subseteq V(S) \tag{b3}$$

$$\forall ([p]_S \mapsto [q]_T) \in f. [q]_T = [p]_S \tag{b1$^+$}$$

$$\forall ([p]_S \mapsto [q]_T) \in f. \langle a, b\rangle \in [q]_T \setminus [p]_S \Rightarrow \{a, b\} \cap V(S) \neq \emptyset \tag{b2$^+$}$$

$$\forall ([p]_S \mapsto [q]_T) \in f. \langle a, b\rangle \in [q]_T \setminus [p]_S \Rightarrow \{a, b\} \not\subseteq V(S) \tag{b3$^+$}$$

Proof. We show only the correspondence between restricted p1-maps and b1-maps, the others are similar.

p1 ⇒ b1) Suppose $h : S \to T$ is a restricted p1-map. By Lemma 5 there exists a b-map $f : [S] \to [T]$. We need to show that f satisfies bound (b1). Since h is a restricted p1-map, there is no triple $\langle a, p, b\rangle \in T \setminus h(S)$, by which it follows that there is no $[h(p)]_T \in [T] \setminus [h(S)]$. From Lemma 5 we have $f([p]_S) = [h(p)]_T$ for all $[p]_T \in [S]$, so there can be no $f([p]_S) \in [T] \setminus [h(S)]$, add the fact that f_h is an inclusion map, and get $f([p]_S) = [p]_S$ for all $[p] \in [S]$.

p1 ⇐ b1) Suppose $f : [S] \to [T]$ is b1-map. By Lemma 4 f induces an restricted p-map $h_f : S \to T$. It suffices to show that h_f satisfies the (p1) bound. By the definition of h_f we have, for all $p \in E(S)$, that $f([p]_S) = [h_f(p)]_T$, whence, since f satisfies (b1), it follows that $[p]_S = [h_f(p)]_T$, and there can be no triple $\langle a, h(p), b\rangle \in T \setminus h(S)$. □

The notion of limited generalisation introduced prior to Example 5 can now be formalised in terms of binary factors as injectivity of bounded b-maps. Edges from the source may collapse in the target only when they relate the same data elements:

Proposition 5. *Let $f : [S] \to [T]$ be an inclusion map. Then f is bound* b*-map of S to T only if f is injective.*

Proof. Suppose that f is a b3-map, and suppose for reduction ad absurdum that f is not injective. Then there are distinct $[p]_S, [q]_S \in [S]$ such that $h([p]_S) = h([q]_S)$. Since $[p]_S \neq [q]_S$, there is some $\beta := \langle a, b\rangle$ such that $\beta \in [p]_S$ and $\beta \notin [q]_S$ (or vice versa, the other case is analogue). Given that f is an inclusion map, $\beta \in f([p]_S)$ and, by assumption, $\beta \in f([q]_S)$. However, the fact that $\beta \notin [p]_S$ and $\{a, b\} \in V(S)$ violates the (b3) bound—a contradiction. □

We repeat the claim for restricted p-maps:

Proposition 6. *If $h : S \to T$ is a bound and restricted* p*-map, then for $p, q \in E(S)$, $h(p) = h(q)$ only if $[p]_S = [q]_S$.*

Proof. Let h be a bound and restricted p-map of an RDF tripleset S. Assume $h(p) = h(q)$, but $[p]_S \neq [q]_S$ for some $p, q \in E(S)$. Then there is a $\beta := \langle a, b\rangle$ such that $\beta \in [p]_S$ and $\beta \notin [q]_S$ (or vise verse, this case is analogue). Since $a, b \in V(S)$ and $\beta \notin [q]_S$ this breaks bound (p3), so h cannot be bound. □

3.1 Computing p-maps

In [19] we have given an algorithm which in polynomial time in the size of the input RDF triplesets computes a restricted pn-map between two triplesets assuming RDF tripleset representation. Here, listed in Algorithm 1, we repeat the same algorithm, but use the binary factor representation in order to show how well it lends itself to computation of such maps.

```
Input  : RDF triplesets $S$ and $T$, bound pn.
Output: $\top$ if there is a pn-map $S \to T$, or $\bot$ if none
        exists.
for $[p]_S \in [S]$
    if $p \in V(S)$
        if $[p]_S \subseteq [p]_T$
            if not SatBound($[p]_T \setminus [p]_S$, pn) return $\bot$;
        else return $\bot$;
    else
        bool found := false;
        for $[q]_T \in [T]$
            if not found and $[p]_S \subseteq [q]_T$
                if not SatBound($[q]_T \setminus [p]_S$, pn)  break;
                found := true;
        if not found return $\bot$;
return $\top$;
begin func SatBound($\beta$, pn)
    if (pn = p1 and $\beta \neq \emptyset$) or
       (pn = p2 and $\forall a, b(\langle a, b\rangle \in \beta \to a, b \notin V(S))$) or
       (pn = p3 and $\forall a, b(\langle a, b\rangle \in \beta \to a, b \not\subseteq V(S))$)
        return false;
    else
        return true;
```

Algorithm 1: Checks the existence of a pn-map of RDF tripleset S to RDF tripleset T.

The algorithm is put to use in a prototype implementation available online at `http://sws.ifi.uio.no/MapperDan/`. Mapper Dan takes two RDF files—a source and a target—as input and computes all restricted p-maps of the source to the target, and, if the user chose to specify one, the maps must satisfy a set of bounds given for a portion or all of the edges in the source. If no such restricted p-map exists, Mapper Dan will suggest a weakening of bounds for edges necessary to have a valid map of the given source and target. The map may then be applied to the source and merged with the target, be used to produce a SPARQL construct query which represents the map, or to rewrite a SPARQL select query over the source vocabulary to a query over the target vocabulary.

4. SUMMARY AND FUTURE WORK

This paper has introduced a restricted form of standard graph homomorphisms applied to RDF triplesets which are especially suited for conservative transformations of the data content in such representations, and called them p-maps. These maps appear in their most powerful form in pair with bounds which restrain the data content of the target such that the transformed data from the source is not distorted by a transformation into the target. Bounded maps also have the prominent feature of being preserved under composition and can thus be used to make sure that the original data is faithfully represented in targets through multiple transformations. We have shown how the different bounds on p-maps can interrelate to form a set of transformation tools with which one can exercise differentiated control over RDF data transformations. Lastly, we translated the notion of restricted p-maps to b-maps to reveal different aspects of these maps and to enable easy computational descriptions of restricted p-maps, and given an algorithm to decide existence of bounded and restricted p-maps of RDF triplesets.

The main purpose of this exposition has been to explore the characteristics of the different bounds on RDF data transformations. These transformations and bounds are simple and intuitive, yet they are powerful and useful within the Linked Data realm. Nevertheless, their simplicity also introduce many interesting open issues. The current triple-to-triple transformations are in many cases too restrictive, disallowing transformations of more complex patterns, for instance maps where triples are mapped to *chains* of triples—such a mapping is illustrated in Example 12. In [19, 17, 18] we establish results for and report successful usage of a restrictive variant of chain-to-chain maps geared for SPARQL construct queries, mapping chains in the `WHERE` block to chains in the `SELECT` block of the query and show how this establishes a "p-map relationship" from the parts of RDF tripleset to which the query is applied and the results of the construct query. However, the exact formulation of maps of more complex structures for RDF triplesets, their characteristics, and the treatment of blank nodes by such maps, remains an open issue.

Example 12. *Compound data is often represented in RDF by clustering the different singular data items round a common node, e.g., as for address data: street address, zip code and country may be grouped by a blank node which is typed as an address. The following illustrates the case. Assume we have the following two representations of the same address data:*

```
:Slottet  ex:street-address  "Henrik Ibsens gate 1" ;
          ex:country         "Norway" .
```

```
:Slottet  vcard:adr
   [ vcard:street-address  "Henrik Ibsens gate 1" ;
     vcard:country-name    "Norway"  ] .
```

There is mapping from the first to the second representation by sending `ex:street-address` *to the chain* `vcard:adr`, vcard:street-address, *and* ex:country *to* vcard:adr, vcard:country-name, *but the theory developed in this paper does not support this.*

In their current formulation, p-maps do not care about the semantics of their source or target, they are only concerned with the graph-like structure of the triplesets. While this is intentional—giving attention to RDF transformations without resorting to schema definitions and reasoning—allowing our maps to exploit simple semantics, for example partitioning vertices into vocabulary elements and data elements and possibly treating these sets differently, would indeed increase the practical value of such maps.

An immediate further development is to produce algorithms for all types of p-maps, and to extend and improve the functionality of Mapper Dan. It would also be interesting to see how p-maps look expressed in terms of *directed, labelled multigraphs* and *directed, labelled hypergraphs* as is used in [2], and also possibly *bipartite graphs* [8], and see if these representations of RDF graphs reveal new natural bounds on p-maps or allow a more abstract formulation of existing bounds. A thorough exposition of how p-maps relate to data exchange [5, 13] and data integration [14] would also be a fascinating read.

5. REFERENCES

[1] M. Arenas, C. Gutierrez, and J. Pérez. Foundations of RDF Databases. In S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M.-C. Rousset, and R. A. Schmidt, editors, *Reasoning Web*, volume 5689 of *LNCS*, pages 158–204. Springer, 2009.

[2] J.-F. Baget. RDF Entailment as a Graph Homomorphism. In *Proc. of the 4th Int. Semantic Web Conference*, volume 3729 of *LNCS*, pages 82–96. Springer, 2005.

[3] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 41(1), 2008.

[4] D. Brickley and L. Miller. FOAF Vocabulary Specification 0.98, 2010. http://xmlns.com/foaf/spec/.

[5] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336:89–124, 2005.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[7] 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.

[8] J. Hayes and C. Gutierrez. Bipartite Graphs as Intermediate Model for RDF. In *Proc. of the 3th Int. Semantic Web Conference*, volume 3298 of *LNCS*, pages 47–61. Springer, 2004.

[9] P. Hayes. RDF Semantics. W3C Recommendation, W3C, 2004.

[10] P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford University Press, 2004.

[11] D. Hutter. Some Remarks on the Annotation %cons, 1999.

[12] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, W3C, 2004.

[13] P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In C. Li, editor, *PODS*, pages 61–75. ACM, 2005.

[14] M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[15] D. C. Makinson. Logical Friendliness and Sympathy in Logic. In *Logica Universalis*. Birkhauser Basel, 2005.

[16] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, W3C, 2008.

[17] A. Stolpe and M. G. Skjæveland. Conservative Repurposing of RDF Data. 10th Int. Semantic Web Conference, 2011. Poster paper.

[18] A. Stolpe and M. G. Skjæveland. From Spreadsheets to 5-star Linked Data in the Cultural Heritage Domain: A Case Study of the Yellow List. In *Norsk informatikkonferanse (NIK 2011)*, pages 13–24. Tapir, 2011.

[19] A. Stolpe and M. G. Skjæveland. Preserving Information Content in RDF Using Bounded Homomorphisms. In E. Simperl, P. Cimiano, A. Polleres, Ó. Corcho, and V. Presutti, editors, *ESWC*, volume 7295 of *LNCS*, pages 72–86. Springer, 2012.

[20] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle. SWRC Ontology, 2005. http://ontoware.org/swrc/.