Compact Summaries of Rich Heterogeneous Graphs

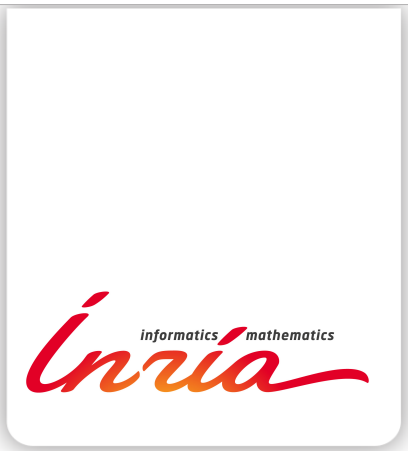
Šejla Čebirić, François Goasdoué, Pawel Guzewicz, Ioana Manolescu

Compact Summaries of Rich Heterogeneous Graphs

Šejla Čebirić, François Goasdoué, Paweł Guzewicz, Ioana Manolescu

RESEARCH REPORT
N° 8920
June 2016
Project-Teams CEDAR

ISSN 0249-6399 ISRN INRIA/RR--8920--FR+ENG



Compact Summaries of Rich Heterogeneous Graphs

Šejla Čebirić, François Goasdoué, Paweł Guzewicz, Ioana Manolescu

Project-Teams CEDAR

Research Report n° 8920 — version 6 — initial version June 2016 — revised version July 2018 — 40 pages

Abstract:
Large data graphs with complex and heterogeneous structure, possibly featuring typed data and an ontology encoding the application-domain semantics, are widely used nowadays. The literature provides many solutions for building succinct representations of graphs, called summaries, in particular based on graph quotients through an equivalence relation between graph nodes.
We consider efficient and compact summarization of rich heterogeneous graphs, in particular RDF ones, which may feature *data edges*, *typed nodes*, and an *ontology*. First, we devise *new graph node equivalence relations*, particularly tolerant of structural heterogeneity; they lead to compact yet informative quotient summaries. Second, we show how to *extend any node equivalence relation* (including, but not limited to ours) to types and ontologies, and provide the first in-depth study of the *interplay between quotient summarization and RDF graph saturation*, which defines the semantics of an RDF graph, in particular in the presence of an ontology. We establish a sufficient condition on a node equivalence relation, which if met allows an efficient method, called *shortcut*, for summarizing RDF graphs. We describe novel, efficient, incremental algorithms for summarizing graphs with our node equivalence relations, and experiments validating their performance.

Key-words: Reasoning and Knowledge Representation, Databases, RDF Graphs, Semantic Web, Graph Summaries

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Compact Summaries of Rich Heterogeneous Graphs

Résumé : Les grands graphes de données, à structure potentiellement complexe et hétérogène, comprenant parfois des noeuds typés et/ou une ontologie décrivant la sémantique d'un domaine d'application, sont utilisés largement de nos jours. Des nombreuses méthodes pour résumer de tels graphes ont été proposées; chacune aboutit à une structure compacte représentant l'essentiel de l'information sur la structure du graphe. Une famille de telles méthodes est basée sur la construction d'un graphe quotient, à partir d'une relation d'équivalence entre les noeuds du graphe d'origine.

Dans ce travail, nous nous intéressons à la construction efficace de résumés compacts pour des graphes de données hétérogènes, en particulier des graphes RDF, qui peuvent comporter des données, des noeuds typés, et une ontologie. Nous proposons de nouvelles relations d'équivalence entre les noeuds d'un graphes, relations particulièrement adaptées à la hétérogénéité souvent rencontrée dans des graphes RDF. Ces relations conduisent à des résumés quotients qui sont compacts, tout en préservant de l'information sur la structure du graphe. Nous montrons comment tout relation d'équivalence entre les noeuds d'un graphe (comprenant, mais ne se limitant pas aux relations que nous introduisons) peut être étendue aux noeuds comportant des types, ainsi qu'aux ontologies. Par la suite, nous présentons la première étude approfondie de l'interaction entre les résumés par quotient et la saturation d'un graphe RDF, qui définit sa sémantique en présence d'une ontologie. Nous identifions une condition suffisante sur une relation d'équivalence entre des noeuds du graphe, condition qui, lorsqu'elle est satisfaite, conduit à une méthode rapide, appellée *shortcut* (raccourci) pour la construction du résumé d'un graphe RDF. Nous présentons des nouveaux algorithmes efficace, en particulier des algorithmes incrémentaux, pour résumer des graphes par nos nouvelles relations d'équivalence. Enfin, nous présentons des expériences qui valide la performance de nos algorithmes et l'intérêt des résumés étudiés.

Mots-clés : Représentation de connaissances, Bases de Données, RDF, Web sémantique, Résumés de Graphes

Contents

1 Introduction

Large, complex data graphs are everywhere, from social networks to scientific data, biology and bibliography databases etc. *Data graph summaries*, small-size graphs extracted from the data, have been extensively investigated, with two main goals: (i) Speed up graph query processing by deriving an *index* from the summary; (ii) Use the summary *instead* of the graph, for instance, to *allow humans to explore the structure of a new graph through a GUI*, or to decide that a query *lacks answers on the graph* (thus, save the fruitless effort of evaluating it and instead return the empty result without consulting the graph). Numerous graph summaries have been proposed in the past, e.g., [15, 11] (see also Section 9). Among these, many are *graph quotients*: based on an *equivalence notion* $\equiv$ among graph nodes, a quotient summary has a node for each equivalence class. Quotient summaries are easy to interpret by human users, and have interesting properties from a computational perspective (see also Section 3).

The starting point of our work is *quotient summarization of RDF graphs*, the W3C standard for representing Semantic Web data. RDF graphs (and also many non-RDF ones) have a high degree of *structural heterogeneity*: graph nodes may be sources and/or targets of very different sets of properties. Further, RDF allows attaching zero, one or more *types* to nodes; this requires applications to exploit type information when available, but also use graphs partially or completely untyped. Finally, RDF graphs may also include an *ontology* describing relationships between the different node types and edges labels (properties) present in the graph. Ontologies capture crucial domain knowledge about an application, and they allow to *interpret and enrich* the graph based on such knowledge. For instance, an ontology stating that any node with an ISBN number is of type publication, on a graph stating that p has the ISBN i, allows to *infer* that p is a publication. Such *implicit* information crucially contributes to a graph's *semantics*; for instance, if a query q requires "all publications with an ISBN", q would be empty without taking inference into account, whereas the correct answer as defined by the SPARQL query language for RDF graphs should include p.

In the sequel of the paper, Section 2 introduces the graphs we consider, then Section 3 recalls quotient summaries.

Next, we present our contributions:

(i) We propose *two new structural node equivalence relations* (applicable to any labeled, directed graph), based on a novel concept of *data property cliques* (Section 4). Different from known node equivalence relations, which are based on *conjunctions* of structural conditions satisfied by equivalent nodes, our equivalences are based on *disjunctive* conditions. While this makes our summaries less suitable to serve as indexes for conjunctive queries, they are *very tolerant of structural heterogeneity*, thus they are: typically very *compact* (orders of magnitude smaller than the graph), and very suitable for *summary-based graph exploration and visualization*. Indeed, their compactness was an argument of chosing them for integration into LODAtlas [32], an interactive portal for navigating and searching through Linked Open Data.

(ii) When graphs have type edges, we identify two methods for extending *any* node equivalence relation and accordingly, extend *any quotient summarization method* to such edges. We derive concrete

	Any data graphs	RDF data graphs
Any $\equiv$ relation	–	Sections 6.1 and 6.2
Our $\equiv$ relations	Section 4 and 7.1	Sections 5, 6.3 and 7.2

Table 1: Outline of the applicability of our contributions.

summaries for such graphs based on our novel equivalence relations (Section 5).

(iii) When graphs have ontologies, we provide the first *in-depth analysis of the intricate interplay between summarization and saturation*, the reasoning mechanism used to complement an RDF graph with the implicit data it contains (Section 6). Specifically: ($iii.a$) We show that these two operations do not always commute. Next, we identify a *sufficient condition on a node equivalence relation* which enables summarizing *the explicit and implicit* edges of a graph, without materializing the implicit ones, through a method we call *shortcut*. This method may be much faster than the one which infers all implicit edges before summarizing. ($iii.b$) We formally establish that two of our equivalence relations allow the shortcut while two others do not.

(iv) We provide *efficient (linear-time in the size of the graph) algorithms* for building our summaries, which are also *incremental*: an edge added to (or removed from) the graph can be reflected in *constant time* in the summary (under an assumption frequently verified by real-life graphs) (Section 7).

Finally, (v) we experimentally validate the good properties of our algorithms (Section 8). Table 1 recaps the areas in which our contributions apply; the code is available online at `https://team.inria.fr/cedar/projects/rdfsummary/`. We discuss related work in Section 9 then we conclude.

Proofs of our technical results are available in the Appendix.

2 Data graphs

Our work is targeted to *directed graphs, with labeled nodes and edges*. This includes those described in RDF [37], the W3C standard for representing Web data. However, RDF graphs attach special interpretation to certain kinds of edges: (i) *type* edges may be used to attach type information to a data node; (ii) *ontology* edges may describe application-domain knowledge as relationships that hold between edge labels and/or node types.

Below, we introduce the useful terminology for RDF graphs since they are *the most general class of graphs for which our summarization methods apply*. We also isolate significant subsets of such graphs, which will be handled differently during summarization.

An *RDF graph* is a set of *triples* of the form `s p o`. A triple states that its *subject* `s` has the *property* `p`, and the value of that property is the *object* `o`. We consider only well-formed triples, as per the RDF specification [37], using uniform resource identifiers (URIs), typed or untyped literals (constants) and blank nodes (unknown URIs or literals). Blank nodes are essential features of RDF allowing to support *unknown URI/literal tokens*. These are conceptually similar to the labeled nulls or variables used in incomplete relational databases [1], as shown in [14].

Notations. We use `s`, `p`, and `o` as placeholders for subjects, properties and objects, respectively.

Figure 1 (top) shows how to use triples to describe resources, that is, to express class (unary relation) and property (binary relation) assertions. The RDF standard [37] has a set of *built-in classes and properties*, as part of the rdf: and rdfs: pre-defined namespaces. We use these namespaces exactly for these classes and properties, e.g., rdf:type specifies the class(es) to which a resource belongs.

As our running example, Figure 2 shows a sample RDF graph. Black and violet edges encode data and type respectively, e.g., node n_1 has property a whose object (value) is a_1, n_1 is of class (or has type)

Assertion	Triple	Relational notation
Class	s rdf:type o	o(s)
Property	s p o	p(s, o)

Constraint	Triple	OWA interpretation
Subclass	s *subclass* o	$s \subseteq o$
Subproperty	s *subproperty* o	$s \subseteq o$
Domain typing	p *domain* o	$\Pi_{\text{domain}}(s) \subseteq o$
Range typing	p *range* o	$\Pi_{\text{range}}(s) \subseteq o$

Figure 1: RDF (top) & RDFS (bottom) statements.

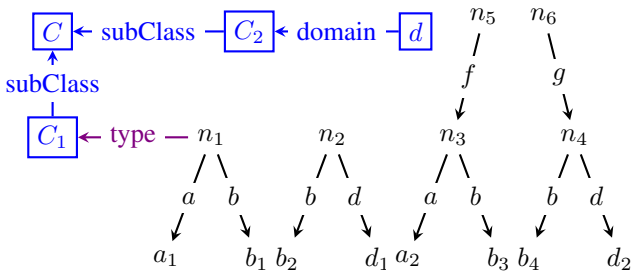


Figure 2: Sample RDF graph $G = \langle D_G, S_G, T_G \rangle$: D_G edges are shown in black, S_G edges in blue, T_G edges in violet.

C_1, etc.

RDF Schema (RDFS). RDFS allows enhancing the descriptions in RDF graphs by declaring *ontological constraints* between the classes and the properties they use. Figure 1 (bottom) shows the four kinds of RDFS constraints, and how to express them through triples. Here, "domain" denotes the first, and "range" the second attribute of every property. In Figure 2, the blue edges connecting boxed nodes are RDFS constraints: they state that C_1 and C_2 are subclasses of C, and that the domain of d is C_2.

RDFS constraints are interpreted under the *open-world assumption (OWA)* [1], i.e., as deductive constraints. *RDF entailment* is the mechanism through which, based on a set of explicit triples and some *entailment rules*, implicit RDF triples are derived. For instance, in Figure 2, node n_1 is of type C_1, which is a subclass of C. Through RDF entailment with the subclass constraint in Figure 1, we obtain the *implicit (entailed) triple* n_1 *type* C stating that n_1 is of class C. Similarly, n_2 and n_4 have property d whose domain is C_2 (thanks to the domain constraint in Figure 1), thus n_2 and n_4 are also of class C_2. Further, because C_2 is a subclass of C, they are also of class C.

In general, a triple s p o is entailed by a graph G, denoted $G \vdash_{\text{RDF}}$ s p o, if and only if there is a sequence of applications of entailment rules that leads from G to s p o (where at each step, triples previously entailed are also taken into account).

RDF graph saturation. Given a set of entailment rules, the *saturation* (a.k.a. closure) of an RDF graph G, is defined as the fixpoint obtained by repeatedly adding to G the triples derived using the entailment rules; we denote it G^∞. For the four constraints shown in Figure 1, which we consider throughout this work, the saturation of G, is finite, unique (up to blank node renaming), and does not contain implicit triples (they have all been made explicit by saturation). An obvious connection holds between the triples entailed by a graph G and its saturation: G entails (leads to, has as logical consequence) the triple s p o if and only if s p o $\in G^\infty$. It is important to note that the semantics of an RDF graph is its saturation [37].

In particular, when querying an RDF graph, the *answer* to the query should be computed both from its explicit and its implicit triples.

For presentation purposes, we may use a *triple-based* or a *graph-based* representation of an RDF graph:

1. Triple-based representation of an RDF graph. We see $\mathtt{G}$ as a *union of three edge-disjoint subgraphs* $\mathtt{G} = \langle \mathtt{D_G}, \mathtt{S_G}, \mathtt{T_G} \rangle$, where: (i) $\mathtt{S_G}$, the **schema** component, is the set of all $\mathtt{G}$ triples whose properties are *subclass*, *subproperty*, *domain* or *range*; we depict such triples with blue edges; (ii) $\mathtt{T_G}$, the **type** component, is the set of *type* triples from $\mathtt{G}$; we show them in violet; (iii) $\mathtt{D_G}$, the **data** component, holds all the remaining triples of $\mathtt{G}$; we display them in black. Note that each union of the $\mathtt{D_G}$, $\mathtt{S_G}$, and $\mathtt{T_G}$ components is an RDF graph by itself.

Further, we call *data property* any property $\mathtt{p}$ occurring in $\mathtt{D_G}$, and *data triple* any triple in $\mathtt{D_G}$.

2. The graph-based representation of an RDF graph. As per the RDF specification [37], *the set of nodes* of an RDF graph is the set of subjects and objects of triples in the graph, while its *edges* correspond to its triples. We define three categories of RDF graph nodes: (i) a *class node* is any node whose URI appears as subject or object of a *subclass* triple, or object of a *domain* or *range* or *type* triple; we show them in blue boxes; (ii) a *property node* is any node whose URI appears as subject or object of a *subproperty* triple, or subject of a *domain* or *range* triple, or property of a triple[1]; in Figure 2, d is a property node. We also show them in blue boxes; (iii) a *data node* as any node that is neither a class nor a property node. We show them in black. Note that the sets of class nodes and of property nodes may intersect (indeed, nothing in the RDF specification forbids it). However, data nodes are disjoint from both class and property nodes.

We will rely on the graph, respectively, the triple-based representation when each is most natural for the presentation; accordingly, we may use *triple* or *edge* interchangeably to denote a graph edge.

3 Summarization framework

We recall the classical notion of *graph quotient*, on which many graph summaries, including ours, are based. In particular, we recall quotients based on *bisimilarity*, and show that their very nature makes them ill-suited to summarize heterogeneous graphs.

Graph quotients. Given a data graph $\mathtt{G}$ and an equivalence relation[2] $\equiv$ over the node of $\mathtt{G}$, the quotient of $\mathtt{G}$ by $\equiv$, denoted $\mathtt{G}_{/\equiv}$, is the graph having (i) a node for each equivalence class of $\equiv$ (thus, for each set of equivalent $\mathtt{G}$ nodes); and (ii) for each edge $n_1 \xrightarrow{a} n_2$ in $\mathtt{G}$, an edge $m_1 \xrightarrow{a} m_2$, where m_1, m_2 are the quotient nodes corresponding to the equivalence classes of n_1, n_2 respectively.

Many known graph summaries, e.g., [29, 20, 9, 35, 24, 11, 13] are quotient-based; they differ in their equivalence relations $\equiv$. Quotient summaries have several desirable properties:

Size guarantees By definition, $\mathtt{G}_{/\equiv}$ is guaranteed to have at most as many nodes and edges as $\mathtt{G}$. Some non-quotient summaries, e.g., Dataguides [15], cannot guarantee this.

Property completeness denotes the fact that every property (edge label) from $\mathtt{G}$ is present on summary edges. This is helpful to users approaching a graph dataset for the first time[3]; it is for this scenario precisely that our summaries are used in LODAtlas [32].

[1] A property node must be a *node*, i.e., merely appearing in a property position does not make an URI a property node; for this, the URI needs to appear *as a subject or object* in a triple of the graph.

[2] An equivalence relation $\equiv$ is a binary relation that is reflexive, i.e., $x \equiv x$, symetric, i.e., $x \equiv y \Rightarrow y \equiv x$, and transitive, i.e., $x \equiv y$ and $y \equiv z$ implies $x \equiv z$ for any x, y, z.

[3] Most complete summarization methods, including ours, can be adapted to reflect e.g., only properties above a certain frequency threshold in the graph etc. We will not consider this further.

Structural representativeness is the following property: for any query q that has answers on $\mathtt{G}$, its *structure-only* version q', which copies all the graph patterns of q but erases its possible selections on nodes as well as counting predicates, is guaranteed to have answers on $\mathtt{G}_{/\equiv}$.

For instance, if q_1 is "find all nodes that are target of an f edge and source of a b and a d edge" on the graph in Figure 2, then q_1' is the same as q_1. If the query q_2 is "find all nodes whose labels contain "Alice", having exactly one (not more) incoming f edge and exactly one outgoing b edge", the query q_2' asks for "all nodes having incoming f and outgoing b edges". Thanks to representativeness, quotient summaries can be used to prune empty-answer queries: if $q'(\mathtt{G}_{/\equiv})$ has no answers, then q has no answers on $\mathtt{G}$. Since the summary is often much smaller than $\mathtt{G}$, pruning is very fast and saves useless query evaluation effort on $\mathtt{G}$.

To enjoy the above advantages, in this work, *a summary of* $\mathtt{G}$ *is its quotient through some equivalence relation*.

Two extreme quotient summaries help to see the trade-offs in this context. First, let $\top$ denote the equivalence relation for which all nodes are equivalent: then, $\mathtt{G}_{/\top}$ has a single node with a loop edge to itself for each distinct property in $\mathtt{G}$. This summary collapses (and loses) most of the graph structure. Now, let $\bot$ denote the equivalence relation for which each node is only equivalent to itself. Then, $\mathtt{G}_{/\bot}$ is isomorphic to $\mathtt{G}$ for any graph $\mathtt{G}$; it preserves all the structure but achieves no summarization.

Bisimulation-based summaries. Many known structural quotient summaries, e.g., [29, 9, 21, 12] are based on bisimilarity [19]. Two nodes n_1, n_2 are *forward and/or backward bisimilar* (denoted $\equiv_{\mathtt{fw}}$, $\equiv_{\mathtt{bw}}$ and $\equiv_{\mathtt{fb}}$) iff for every $\mathtt{G}$ edge $n_1 \xrightarrow{a} m_1$, $\mathtt{G}$ also comprises an edge $n_2 \xrightarrow{a} m_2$, such that m_1 and m_2 are also forward and/or backward bisimilar, respectively. The $\equiv_{\mathtt{fw}}$ and $\equiv_{\mathtt{bw}}$ relations only take into account the paths outgoing from (resp. only the paths incoming to) the nodes. The **symmetry** of $\mathtt{G}_{/\mathtt{fb}}$ is an advantage, as it makes it more resistent to minor modeling differences in the data. For instance, let t' be the triple a *hasAuthored* p and t'' be p *hasAuthor* a, which essentially represent the same information. Triple t' would impact the nodes to which a is $\equiv_{/\mathtt{fw}}$, while it would not impact the nodes to which a is $\equiv_{/\mathtt{bw}}$; symetrically, t'' would impact the $\equiv_{/\mathtt{bw}}$ class of p but not its $\equiv_{/\mathtt{fw}}$ class. In contrast, $\equiv_{/\mathtt{fb}}$ reflects this information whether it is modeled in one direction or in the other.

We denote the bisimulation based summaries $\mathtt{G}_{/\mathtt{fw}}$ (forward), $\mathtt{G}_{/\mathtt{bw}}$ (backward) and $\mathtt{G}_{/\mathtt{fb}}$ (forward and backward), respectively. They tend to be **large** because *bisimilarity is rare in heterogeneous data graphs*. For instance, each node of the graph in Figure 2 is only $\equiv_{/\mathtt{fb}}$ to itself, thus $\equiv_{/\mathtt{fb}}$ is useless for summarizing it; our experiments in Section 8 confirm this on many graphs. To mediate this problem, k**-bisimilarity** has been introduced [22], whereas nodes are k-forward (and/or backward) bisimilar iff their adjacent paths *of length at most* k are identical; the smaller k is, the more permisive the equivalence relation.

One drawback of k-bisimilarity is that it requires users to guess the k value leading to the best compromise between compactness (which favors low k; note that $k = 0$ leads exactly to $\mathtt{G}_{/\top}$) and structural information in the summary (high k). Further, *even* 1*-bisimilarity is hard to achieve* in heterogeneous graphs. For instance, Figure 3 shows the $\mathtt{1fb}$ summary of the sample graph in Figure 2[4]. Nodes in the bottom row of $\mathtt{G}$, which only have incoming a, b, respectively d edges, and have no outgoing edges, are summarized together. However, none of n_1, n_2, n_3 and n_4 are equivalent, because of the presence/absence of a and d edges, and of their possible incoming edges.

Any structural equivalence relation cuts a trade-off between compactness and structure preservation. Below, we introduce two relations leading to compact summaries that *in addition* cope well with the data heterogeneity frequently encountered in RDF (and other) graphs.

Other interesting properties of such relations are: the complexity of building the corresponding summary, and of updating it to reflect graph updates. Further, the presence of type and schema (ontology) triples has not been formally studied in quotient graph summarization.

[4]In Figure 3, type and schema triples are summarized according to the method we propose below. However, the treatment of these triples is orthogonal to the interesting aspects of this example.

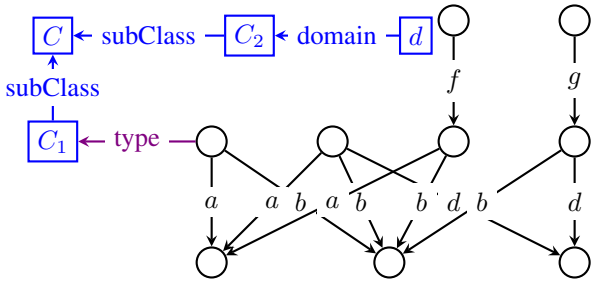


Figure 3: 1fb summary of the RDF graph in Figure 2.

4 Data graph summarization

We first consider graphs made of *data triples* only, thus of the form $\mathtt{G} = \langle \mathtt{D_G}, \emptyset, \emptyset \rangle$. We define the novel notion of *property cliques* in Section 4.1; building on them, we devise new graph node equivalence relations and corresponding graph summaries in Section 4.2. Summarization will be generalized to handle also *type triples* in Section 5, and *type and schema triples* in Section 6.

4.1 Data property cliques

We are interested in defining equivalence relations between two data nodes, that can cope with the heterogeneity present in many real-life data graphs. For example, up to 14 data properties (such as title, author, year, but also note, etc.) are used to describe conference papers in a graph version of the DBLP bibliographic database. Each paper has a certain subset of these 14 properties, and has some of them, e.g., authors, with multiple values; we counted more than 130 such distinct property subsets in a small (8MB) fragment of DBLP. To avoid the "noise" introduced by such structural heterogeneity, we need node equivalence relations that look *beyond* it, and consider that all the nodes corresponding to conference publications are equivalent.

To do that, we first focus on the way data properties (edge labels) are organized in the graph. The simplest relation that may exist between two properties is *co-occurrence*, when a node is the source (or target) of two edges carrying the two labels. However, in heterogeneous RDF graphs such as DBLP, two properties, say author and title, may co-occur on a node n, while another node n' has title, year, and howpublished: we may consider *all* these properties (author, title, year and howpublished) related, as they (directly or *transitively*) co-occur on some nodes. Formally:

Definition 1. (PROPERTY RELATIONS AND CLIQUES) *Let* $\mathtt{p}_1$, $\mathtt{p}_2$ *be two data properties in* G*:*

1. $\mathtt{p}_1, \mathtt{p}_2 \in \mathtt{G}$ *are* source-related *iff either: (i) a data node in* G *is the subject of both* $\mathtt{p}_1$ *and* $\mathtt{p}_2$*, or (ii)* G *holds a data node that is the subject of* $\mathtt{p}_1$ *and a data property* $\mathtt{p}_3$*, with* $\mathtt{p}_3$ *and* $\mathtt{p}_2$ *being source-related.*

2. $\mathtt{p}_1, \mathtt{p}_2 \in \mathtt{G}$ *are* target-related *iff either: (i) a data node in* G *is the object of both* $\mathtt{p}_1$ *and* $\mathtt{p}_2$*, or (ii)* G *holds a data node that is the object of* $\mathtt{p}_1$ *and a data property* $\mathtt{p}_3$*, with* $\mathtt{p}_3$ *and* $\mathtt{p}_2$ *being target-related.*

A maximal set of data properties in G *which are pairwise source-related (respectively, target-related) is called a* source *(respectively,* target*)* property clique.

In the graph in Figure 2, properties a and b are source-related due to n_1 (condition 1. in the definition). Similarly, b and d are source-related due to n_2; consequently, a and d are source-related (condition 2.).

n	n_1	n_2	n_3	n_4	n_5	n_6
$SC(n)$	$\{a, b, d\}$	$\{a, b, d\}$	$\{a, b, d\}$	$\{a, b, d\}$	$\{f\}$	$\{g\}$
$TC(n)$	$\emptyset$	$\emptyset$	$\{f\}$	$\{g\}$	$\emptyset$	$\emptyset$

n	a_1	a_2	b_1	b_2	b_3	d_1	d_2
$SC(n)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$TC(n)$	$\{a\}$	$\{a\}$	$\{b\}$	$\{b\}$	$\{b\}$	$\{d\}$	$\{d\}$

Table 2: Source and target cliques of G nodes (Figure 2).

Thus, a source clique of this graph is $SC_1 = \{a, b, d\}$. Table 2 shows the target and source cliques of all data nodes from Figure 2.

It is easy to see that the set of non-empty source (or target) property cliques is a *partition over the data properties of* G. Further, if a node $n \in$ G is source of some data properties, they are all in the same source clique; similarly, all the properties of which n is a target are in the same target clique. This allows us to refer to *the source (or target) clique of* n, denoted $SC(n)$ and $TC(n)$.

4.2 Strong and weak node equivalences

Building on property cliques, we define two main *node equivalence relations* among the data nodes of a graph G:

Definition 2. (STRONG EQUIVALENCE) *Two data nodes of* G *are* strongly equivalent*, denoted* $n_1 \equiv_S n_2$*, iff they have the same source* and *target cliques.*

Strongly equivalent nodes have the same structure of *incoming and outgoing* edges. In Figure 2, nodes n_1, n_2 are *strongly* equivalent to each other, and so are e.g., a_1, a_2, b_1, b_2 and b_3 etc.

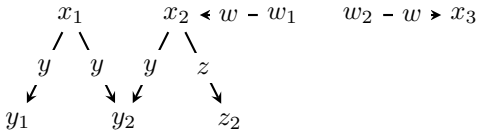


Figure 4: Sample weakly equivalent nodes: x_1, x_2, x_3.

A second, weaker notion of node equivalence could request only that equivalent nodes share the same *incoming or outgoing* structure, i.e., they share the same source clique or the same target clique. Figure 4 illustrates this. Nodes x_1, x_2 have the same source clique because they both have outgoing y edges. Further, x_2 and x_3 have the same target clique because both have incoming w edges. Since equivalence must be transitive, it follows that x_1 and x_3 must also be considered weakly equivalent, since they "follow the same pattern" of having at least one incoming w edge, or at least one outgoing y or z edge, and no other kinds of edges. Formally:

Definition 3. (WEAK EQUIVALENCE) *Two data nodes are* weakly equivalent*, denoted* $n_1 \equiv_W n_2$*, iff:* (i) *they have the same non-empty source or non-empty target clique, or* (ii) *they both have empty source and empty target cliques, or* (iii) *they are both weakly equivalent to another node of* G*.*

It is easy to see that $\equiv_W$ and $\equiv_S$ are equivalence relations and that strong equivalence implies weak equivalence.

In Figure 2, $n_1, \ldots, n_4$ are *weakly* equivalent to each other due to their common source clique SC_1; a_1, a_2 are weakly equivalent due to their common target clique etc.

4.3 Weak and strong summarization

Notation: representation function. We say the summary node of $\mathtt{G}_{/\equiv}$ corresponding to the equivalence class of a $\mathtt{G}$ node n represents n, and denote it $f_{\equiv}(n)$ or simply $f(n)$ when this does not cause confusion. We call $f_{\equiv}$ the *representation function* of the equivalence relation $\equiv$ over $\mathtt{G}$.

Weak summarization. The first summary we define is based on weak equivalence:

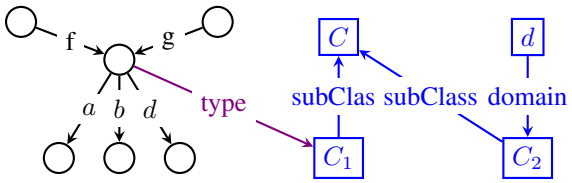


Figure 5: Weak summary of the RDF graph in Figure 2.

Definition 4. (WEAK SUMMARY) *The* weak summary *of a data graph* $\mathtt{G}$*, denoted* $\mathtt{G}_{/\mathtt{W}}$*, is its quotient graph w.r.t. the weak equivalence relation* $\equiv_{\mathtt{W}}$.

The weak summary of the graph in Figure 2 is depicted by the black nodes and edges in Figure 5[5]; summary nodes are shown as unlabeled circles, to denote that they are anonymous (new) nodes, each of which represents one or more $\mathtt{G}$ nodes. The central one represents n_1, n_2, n_3 and n_4. Its outgoing edges go towards nodes representing, respectively, a_1 and a_2; b_1, b_2 and b_3; finally, d_1 and d_2. Its incoming edges come from the representative of n_5 (which was a source of an f edge in $\mathtt{G}$), respectively from the representative of n_6 (source of g).

The weak summary has the following important property:

Proposition 1. (UNIQUE DATA PROPERTIES) *Each* $\mathtt{G}$ *data property appears exactly once in* $\mathtt{G}_{/\mathtt{W}}$.

Importantly, the above Proposition 1 warrants that $|\mathtt{G}_{/\mathtt{W}}|$, the number of edges in $\mathtt{G}_{/\mathtt{W}}$, is exactly the number of distinct data properties in $\mathtt{G}$. This observation is used in our weak summarization algorithms (Section 7). By definition of a quotient summary (Section 3), *this is the smallest number of edges a summary may have* (since it has at least one edge per each distinct property in $\mathtt{G}$). Thus, $\mathtt{G}_{/\mathtt{W}}$ is a minimal-size quotient summary (like $\mathtt{G}_{/\top}$ from Section 3, but much more informative than it). As our experiments show, $|\mathtt{G}_{/\mathtt{W}}|$ is typically 3 to 6 orders of magnitude smaller than $|\mathtt{G}|$.

Strong summarization. Next, we introduce:

Definition 5. (STRONG SUMMARY) *The* strong summary *of the graph* $\mathtt{G}$*, denoted* $\mathtt{G}_{/\mathtt{S}}$*, is its quotient graph w.r.t. the strong equivalence relation* $\equiv_{\mathtt{S}}$.

The strong summary of the graph of Figure 2 is shown by the black edges and nodes in Figure 6. Similarly to the weak summary (Figure 5), the strong one features a single node source of a, respectively, b, d, f and g edges. However, differently from $\mathtt{G}_{/\mathtt{W}}$, the strong summary splits the data nodes whose source clique is $\{a, b, d\}$ in *three equivalence classes:* n_1 and n_2 have the empty target clique, while that of n_2 is $\{f\}$ and that of n_3 is $\{g\}$. Thus, two data nodes represented by the same strong summary node have similar structure both in their inputs and outputs; in contrast, a weak summary (recall Figure 4)

[5]The violet and blue edges serve our discussion later on.

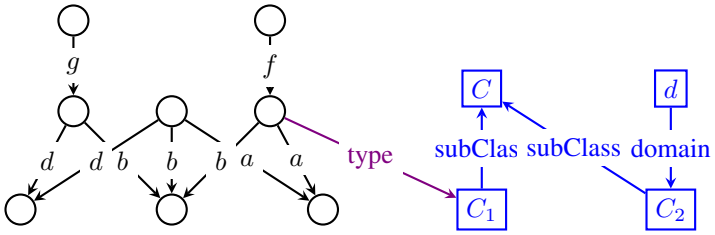


Figure 6: Strong summary of the RDF graph in Figure 2.

represents together nodes having similar structure in their inputs *or* outputs, *or* which are both equivalent to another common node. As we can see, *strong summarization leads to finer-granularity* summaries. An effect of this finer granularity is that in $\mathtt{G}_{/\mathtt{S}}$, several edges may have the same label, e.g., there are three edges labeled b in Figure 5 (whereas for $\mathtt{G}_{/\mathtt{W}}$, as stated in Proposition 1, this is not possible). Our experiments (Section 8) show that while $\mathtt{G}_{/\mathtt{S}}$ is often somehow larger than $\mathtt{G}_{/\mathtt{W}}$, it still remains many orders of magnitude smaller than the original graph.

By definition of $\equiv_{\mathtt{S}}$, equivalent nodes have the same source clique and the same target clique. This leads to:

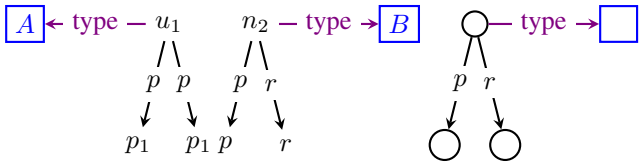
Proposition 2. (STRONG SUMMARY NODES AND G CLIQUES) $\mathtt{G}_{/\mathtt{S}}$ *has exactly one node for each source clique and target clique of a same node* $n \in \mathtt{D}_{\mathtt{G}}$.

Proposition 2 is exploited by the implementations of our strong summarization algorithms (Section 7).

5 Typed data graph summarization

We generalize our approach to summarize graphs with data and type triples, thus of the form $\mathtt{G} = \langle \mathtt{D}_{\mathtt{G}}, \mathtt{T}_{\mathtt{G}}, \emptyset \rangle$.

Starting from an equivalence relation $\equiv$ defined over *data* nodes, in order to summarize $\mathtt{D}_{\mathtt{G}} \cup \mathtt{T}_{\mathtt{G}}$, two questions must be answered: (i) how should $\equiv$ be extended on class nodes (such as C_1 in Figure 2)? and (ii) how should the type edge(s) of a node n be taken into account when determining to whom n is equivalent? Below, we answer these questions for *any* equivalence relation $\equiv$, then instantiate our answer to the weak and strong relations we defined.



To study the first question, consider the sample graph above, and a possible summary of this graph at its right. Assume that the types A and B are considered equivalent. Quotient summarization represents them both by the summary node at the top right, which (like all summary nodes) is a "new" node, i.e., it is neither A nor B. Observe that this summary *compromises representativeness* for queries over both the data and the type triples: for instance, the query asking for "nodes of type A having property r" is empty on the summary (as type A has been conflated with type B) while it is non empty on the graph.

To avoid this, we argue that when moving from data to typed data graphs, any equivalence relation $\equiv$ between data nodes should be extended to class nodes as follows: 1. *any class node is only equivalent*

to itself and 2. *any class node is only represented by itself*, hence a graph has the same class nodes as its summary.

We now consider the second question: how should $\equiv$ be extended to exploit not only the data but also the type triples? Note that two nodes may have similar incoming/outgoing data edges but different type edges, or vice-versa, the same types but very different data edges. We introduce two main alternatives below, then decline them for weak and strong summarization.

5.1 Data-then-type summarization

This approach consists of using an equivalence $\equiv$ defined based on data properties in order to determine which data nodes are equivalent and thus to build summary nodes, and data edges between them. Afterward, for each triple n *type* C in $\mathtt{G}$, add to $\mathtt{G}_{\equiv}$ a triple $f_{\equiv}(n)$ *type* C, where we recall that $f_{\equiv}(n)$ is the representative of n in $\mathtt{G}_{\equiv}$. This approach is interesting, e.g., when only some of the nodes have types (often the case in RDF graphs). In such cases, it makes sense to first group nodes according to their data edges, while still preserving the (partial) type information they have. *We extend the* $\mathtt{W}$, *respectively* $\mathtt{S}$ *summaries to type triples, by stating that they* (i) *represent each class node by itself; and* (ii) *follow a data-then-type approach*, as described above.

In Figure 5, the black and violet edges (including the C_1 node) depict the weak summary of the black and violet graph triples Figure 2. The type edge reads as: *at least one* of the nodes represented by its source, was declared of type C_1 in the input graph. Similarly, the black and violet edges in Figure 6 show the strong summary of the same subset of our sample graph.

To recap, in data-then-type summarization using $\equiv$, two data nodes are represented together iff they are $\equiv$ based on their incoming and outgoing data edges, while a class node is only equivalent to itself (and always represented by itself).

One more point needs to be settled. Some $\mathtt{T}_{\mathtt{G}}$ nodes may have types, but no incoming or outgoing data properties. Strong summarization represents all such nodes together, based on their $(\emptyset, \emptyset)$ pair of source and target cliques. For completeness, we extend weak summaries to also represent such nodes together, by a single special node denoted $N_{\emptyset}$.

5.2 Type-then-data summarization

This approach takes the opposite view that node types are more important when deciding whether nodes are equivalent. Observe that our framework (just like RDF) does not prevent a node from having *several* types. At the same time, representing a node by *each* of its types separately would violate the quotient summarization framework, because a quotient, by definition, represents each node exactly once. Thus, in type-then-data summarization, we extend a given equivalence relation $\equiv$ (based on data properties alone) as follows.

Definition 6. (TYPED EQUIVALENCE) Typed equivalence*, denoted* $\equiv_{\mathtt{T}}$*, is an equivalence relation over* $\mathtt{D}_{\mathtt{G}} \cup \mathtt{T}_{\mathtt{G}}$ *defined as follows: two data nodes* n_1 *and* n_2 *are type-equivalent, noted* $n_1 \equiv_{\mathtt{T}} n_2$*, iff they have exactly the same set of types in* $\mathtt{G}$*, which is non-empty; any class node is only equivalent to itself.*

Intuitively, typed equivalence performs a first-cut data node classification, *according to their sets of types*. In particular, all untyped nodes are equivalent to themselves. This enables the definition of type-then-data summaries as *double quotients*: first, quotient $\mathtt{G}$ by $\equiv_{\mathtt{T}}$; then, quotient the resulting graph by some data node equivalence *only on untyped nodes* (each left alone in an equivalence class of $\equiv_{\mathtt{T}}$), to group them according to their data edges.

Applied to weak summarization, this approach leads to:

Definition 7. (TYPED WEAK SUMMARY) *Let* $\equiv_{\mathtt{UW}}$ *(*untyped weak equivalence*) be an equivalence relation that holds between two data nodes* n_1, n_2 *iff* (i) n_1, n_2 *have no types in* $\mathtt{G}$ *and* (ii) $n_1 \equiv_{\mathtt{W}} n_2$*. The* typed

weak summary $\mathtt{G}_{/\mathtt{TW}}$ *of a graph* $\mathtt{G}$ *is the untyped-weak summary of the type-based summary of* $\mathtt{G}$*, namely* $(\mathtt{G}_{/\mathtt{T}})_{/\mathtt{UW}}$.

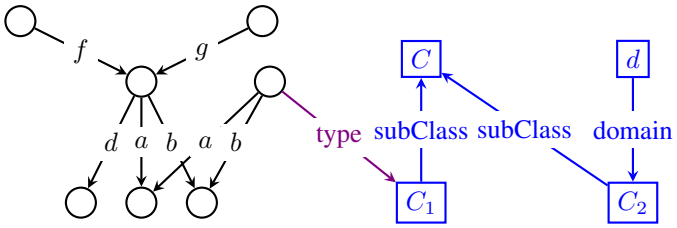


Figure 7: Typed weak summary of the graph in Figure 2.

In Figure 7, the black and violet edges depict the typed weak summary of the data and type edges of the sample graph in Figure 2. Unlike $\mathtt{G}_{/\mathtt{W}}$ (Figure 5), $\mathtt{G}_{/\mathtt{TW}}$ represents the node of type C_1 separately from the untyped ones having similar properties. This reflects the primordial role of types in type-then-data summarization.

In a similar manner, we define:

Definition 8. (TYPED STRONG SUMMARY) *Let* $\equiv_{\mathtt{US}}$ (untyped strong equivalence*) be an equivalence relation that holds between two data nodes* n_1, n_2 *iff (i)* n_1, n_2 *have no types in* $\mathtt{G}$ *and (ii)* $n_1 \equiv_{\mathtt{S}} n_2$*. The* typed strong summary $\mathtt{G}_{/\mathtt{TS}}$ *of an RDF graph* $\mathtt{G}$ *is defined as:* $(\mathtt{G}_{/\mathtt{T}})_{/\mathtt{US}}$.

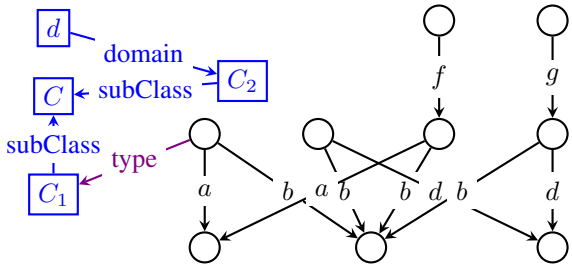


Figure 8: Typed strong summary of the graph in Figure 2.

In Figure 8, the black and violet edges depict the typed strong summary of the data and type edges of the sample graph in Figure 2. Unlike $\mathtt{G}_{/\mathtt{S}}$ (Figure 6), $\mathtt{G}_{/\mathtt{TS}}$ represents the node n_1 of type C_1 separately from n_2, which has no types.

6 RDF graph summarization

We consider now the summarization of general RDF graphs which may also have schema triples, i.e., of the form $\mathtt{G} = \langle \mathtt{D}_{\mathtt{G}}, \mathtt{T}_{\mathtt{G}}, \mathtt{S}_{\mathtt{G}} \rangle$.

6.1 Extending summarization

First, how to extend an equivalence relation $\equiv$ defined on data nodes (and extended as we discussed in Section 5 to class nodes, each of which is only equivalent to itself), to also cover *property nodes*, such as the boxed d node in Figure 2? Such property nodes provide important schema (ontology) information,

which describes how the properties and types present in the data relate to each other, and leads to implicit triples (recall Section 2). For the summary to preserve the semantics of the input graph, by an argument similar to the one at the beginning of Section 5, we impose that $\equiv$ be extended so that *any property node should (also) be equivalent only to itself*, and propose that in any summary, *any property node should be represented by itself*. As a consequence, since any class or schema node is equivalent only to itself and represented only by itself:

Proposition 3. (SCHEMA PRESERVATION THROUGH SUMMARIZATION) *For any equivalent relation $\equiv$ defined on data nodes and extended as specified above to class and property nodes, and any graph $\mathtt{G} = \langle \mathtt{D_G}, \mathtt{T_G}, \mathtt{S_G} \rangle$, it holds that: $\mathtt{S}_{\mathtt{G}_{\equiv}} = \mathtt{S_G}$, that is: the summary of $\mathtt{G}$ through $\equiv$ has exactly the schema triples of $\mathtt{G}$.*

This decision allows us to simply copy schema triples from the input graph to each of its summaries. Figures 5, 6, 7 and 8, considered in their entirety, show respectively full $\mathtt{G}_{/\mathtt{W}}$, $\mathtt{G}_{/\mathtt{S}}$, $\mathtt{G}_{/\mathtt{TW}}$ and $\mathtt{G}_{/\mathtt{TS}}$ summaries of the sample RDF graph in Figure 2.

6.2 Summarization versus saturation

As we explained in Section 2, the semantics of a graph $\mathtt{G}$ includes its explicit triples, but also its *implicit* triples which are not in $\mathtt{G}$, but hold in $\mathtt{G}^{\infty}$ due to ontological triples (such as the triples n_2 *type* C_2 and n_2 *type* C discussed in Section 2).

A first interesting question, then, is: how does saturation impact the summary of a graph? As Figure 1 shows, saturation adds data and type triples. Other entailment rules (see [14]) also generate schema triples, e.g., if C' is a subclass of C'' and C'' is a subclass of C''', then C' is also a subclass of C''' etc. Due to these extra edges, in general, $(\mathtt{G}^{\infty})_{/\equiv}$ and $\mathtt{G}_{/\equiv}$ are different. On one hand, their nodes may be different, but this is not the most interesting aspect, as summary nodes are just "representatives", i.e., they are labeled in a somehow arbitrary fashion. On the other hand (and this is much more meaningful), their graph structure may be different, as it results from graphs with different edges. To separate the mere difference of node IDs from the meaningful difference of graph structure, we define:

Definition 9. (STRONG ISOMORPHISM $\widehat{=}$) *A* strong isomorphism between two RDF graphs $\mathtt{G}_1, \mathtt{G}_2$, *noted $\mathtt{G}_1 \mathrel{\widehat{=}} \mathtt{G}_2$, is a graph isomorphism which is the identity for the class and property nodes.*

Intuitively, strongly isomorphic graphs (in particular, summaries) represent exactly the same information, while the identifiers of their non-class, non-property nodes (shown as unlabeled circles in our examples) may differ.

Next, one could wonder whether saturation *commutes with* summarization, that is, does $(\mathtt{G}^{\infty})_{/\equiv} \mathrel{\widehat{=}} (\mathtt{G}_{/\equiv})^{\infty}$ hold? If this was the case, it would lead to a likely more efficient method for computing the summary of $\mathtt{G}$'s full semantics, that is $(\mathtt{G}^{\infty})_{/\equiv}$, without saturating $\mathtt{G}$ (thus, without materializing all its implicit triples); instead, we would summarize $\mathtt{G}$ and then saturate the resulting (usually much smaller) graph. Unfortunately, Figure 9 shows that this is not always the case. For a given graph $\mathtt{G}$, it traces (top row) its weak summary $\mathtt{G}_{/\mathtt{W}}$ and its saturation $(\mathtt{G}_{/\mathtt{W}})^{\infty}$, whereas the bottom row shows $\mathtt{G}^{\infty}$ and its summary $(\mathtt{G}^{\infty})_{/\mathtt{W}}$. Here, saturation leads to b edges outgoing both r_1 and r_2 which makes them equivalent. In contrast, summarization *before* saturation represents them separately; saturating the summary cannot revert this decision, to unify them as in $(\mathtt{G}^{\infty})_{/\mathtt{W}}$ (recall from Section 2 that saturation can only add edges between $\mathtt{G}$ nodes).

While $(\mathtt{G}^{\infty})_{/\equiv}$ and $(\mathtt{G}_{/\equiv})^{\infty}$ are *not* strongly isomorphic in general, we establish that they *always* relate as follows:

Theorem 1 (Summarization Homomorphism)**.** *Let $\mathtt{G}$ be an RDF graph, $\mathtt{G}_{/\equiv}$ its summary and f the corresponding representation function from $\mathtt{G}$ nodes to $\mathtt{G}_{/\equiv}$ nodes. Then f defines a homomorphism from $\mathtt{G}^{\infty}$ to $(\mathtt{G}_{/\equiv})^{\infty}$.*

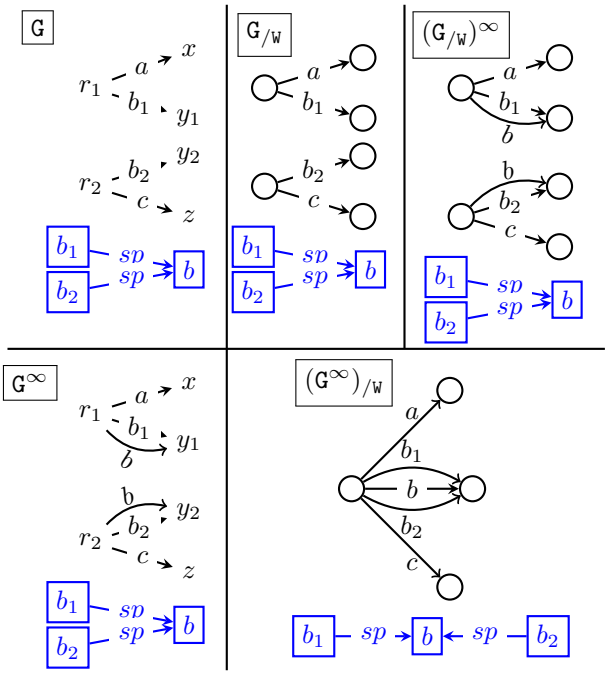


Figure 9: Saturation and summarization on a sample graph.

Since we established that $(\mathtt{G}_{/\equiv})^{\infty}$ is homomorphic to $\mathtt{G}^{\infty}$, would *their* summaries be the same? In other words: are $(\mathtt{G}^{\infty})_{/\equiv}$ and $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ strongly isomorphic?

It turns out that this may hold or not depending on the RDF equivalence relation under consideration. When it holds, we call *shortcut* the following three-step transformation aiming at obtaining $(\mathtt{G}^{\infty})_{/\equiv}$: first summarize $\mathtt{G}$; then saturate its summary; finally, summarize it again in order to build $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$:

Definition 10. (SHORTCUT) *We say* the shortcut holds *for a given RDF node equivalence relation $\equiv$ iff for any RDF graph $\mathtt{G}$, $(\mathtt{G}^{\infty})_{/\equiv}$ and $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ are strongly isomorphic.*

Next, we establish one of our main contributions: a *sufficient condition* under which for *any* quotient summary based on an equivalence relation $\equiv$ as discussed above (where class and property nodes are preserved by summarization), the shortcut holds. In particular, as we will demonstrate (Section 8), the existence of the shortcut can lead to computing $(\mathtt{G}^{\infty})_{/\equiv}$ *substantially faster*.

Theorem 2 (Sufficient shortcut condition)**.** *Let $\mathtt{G}_{/\equiv}$ be a summary of $\mathtt{G}$ through $\equiv$ and $f_{\equiv}$ the corresponding representation function from $\mathtt{G}$ nodes to $\mathtt{G}_{/\equiv}$ nodes (see Figure 10).*

If $\equiv$ satisfies: for any RDF graph $\mathtt{G}$ and any pair (n_1, n_2) of $\mathtt{G}$ nodes, $n_1 \equiv n_2$ in $\mathtt{G}^{\infty}$ iff $f(n_1) \equiv f(n_2)$ in $(\mathtt{G}_{/\equiv})^{\infty}$, then the shortcut holds for $\equiv$.

Figure 10 depicts the relationships between an RDF graph $\mathtt{G}$, its saturation $\mathtt{G}^{\infty}$ and summarization $(\mathtt{G}^{\infty})_{/\equiv}$ thereof, and the RDF graphs that appear at each step of the shortcut computation. The **intuition for the sufficient condition** is the following. On any path in Figure 10, saturation adds edges to its input graph, while summarization "fuses" nodes into common representatives. On the regular path from $\mathtt{G}$ to $(\mathtt{G}^{\infty})_{/\equiv}$, edges are added in the first step, and nodes are fused in the second. On the shortcut (green) path, edges are added in the second step, while node are fused in the first and third steps. The two paths starting from $\mathtt{G}$ can reach $\simeq$ results only if $\mathtt{G}$ nodes fused on the shortcut path, are also fused (when summarizing

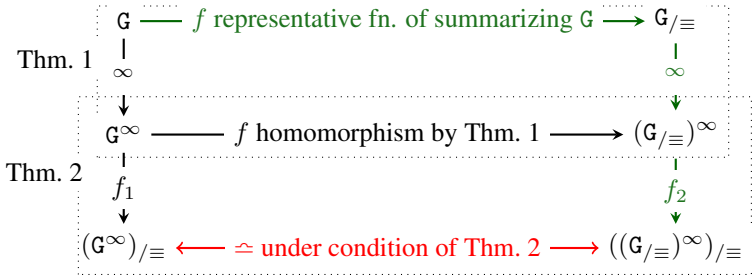


Figure 10: Illustration for Theorem 1 and Theorem 2.

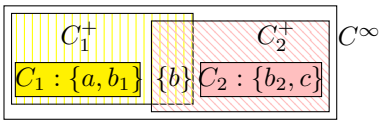


Figure 11: Two source cliques from the graph in Figure 9, their saturations, and their enclosing clique in $\mathtt{G}^\infty$.

$\mathtt{G}^\infty$) on the standard path. In particular, *the first summarization along the shortcut path should not make wrong node fusions*, that is, fusions not made when considering the full $\mathtt{G}^\infty$: such a "hasty" fusion *can never be corrected later on along the shortcut path*, as neither summarization nor saturation split nodes. Thus, an erroneous fusion made in the first summarization step irreversibly prevents the end of the shortcut path from being $\simeq$ to $(\mathtt{G}_{/\equiv})^\infty$. In this case, we remark that $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$ is just homomorphic to $\mathtt{G}^\infty$ (through f then f_2 in Figure 10), hence a compression of it.

6.3 Shortcut results

Next, we investigate for which $\equiv$ does the shortcut hold.

The impact of saturation on property cliques is at the core of our findings. In $\mathtt{G}^\infty$, every $\mathtt{G}$ node has all the data properties it had in $\mathtt{G}$, therefore two data properties belonging to a $\mathtt{G}$ clique are also in the same clique of $\mathtt{G}^\infty$. Further, if the schema of $\mathtt{G}$ comprises *subproperty* constraints, a node may have in $\mathtt{G}^\infty$ a data property that it did not have in $\mathtt{G}$. In turn, this leads to $\mathtt{G}^\infty$ cliques which "subsume and fuse" several $\mathtt{G}$ cliques into one.

For a given clique C of $\mathtt{G}$, we call *saturated clique* and denote C^+ the set of all the properties in C and all their generalizations (superproperties). Observe that C^+ reflects only C and the schema of $\mathtt{G}$; in particular, it does not reflect the data properties shared by nodes in $\mathtt{G}^\infty$, and thus in general C^+ is *not* a clique of $\mathtt{G}^\infty$.

The following Lemma characterizes the relationships between a clique C of $\mathtt{G}$, its saturated version C^+, and the cliques of $\mathtt{G}^\infty$:

Lemma 1 (Saturation vs. property cliques)**.** *Let C, C_1, C_2 be non-empty source (or target) cliques of $\mathtt{G}$.*

1. *There exists exactly one source (resp. target) clique C^∞ of $\mathtt{G}^\infty$ such that $C \subseteq C^\infty$.*
2. *If $C_1^+ \cap C_2^+ \neq \emptyset$, then all the properties in C_1 and C_2 are in the same $\mathtt{G}^\infty$ clique C^∞.*
3. *Any non-empty source (or target) clique C^∞ is a union of the form $C_1^+ \cup \cdots \cup C_k^+$ for some $k \geq 1$, where each C_i is a non-empty source (resp. target) clique of $\mathtt{G}$, and for any C_i, C_j where*

$1 \leq i, j \leq k$ *with* $i \neq j$*, there exist some cliques* $D_1 = C_i, \ldots, D_n = C_j$ *in the set* $\{C_1, \ldots, C_k\}$ *such that:*

$$D_1^+ \cap D_2^+ \neq \emptyset, \ldots, D_{n-1}^+ \cap D_n^+ \neq \emptyset$$

4. *Let* $\mathtt{p}_1, \mathtt{p}_2$ *be two data properties in* $\mathtt{G}$*, whose source (or target) cliques are* C_1 *and* C_2*. Properties* $\mathtt{p}_1, \mathtt{p}_2$ *are in the same source (resp. target) clique* C^∞ *of* $\mathtt{G}^\infty$ *if and only if there exist* k *non-empty source (resp. target) cliques of* $\mathtt{G}$*,* $k \geq 0$*, denoted* $D_1, \ldots, D_k$ *such that:*

$$C_1^+ \cap D_1^+ \neq \emptyset,\ D_1^+ \cap D_2^+ \neq \emptyset, \ldots, D_{k-1}^+ \cap D_k^+ \neq \emptyset, D_k^+ \cap C_2^+ \neq \emptyset.$$

Figure 11 illustrates the lemma for two source cliques of the graph in Figure 9. The source clique $C^\infty = \{a, b_1, b, b_2, c\}$ of the $\mathtt{G}^\infty$ (also shown in Figure 9) encloses the cliques C_1, C_2 of $\mathtt{G}$ (Lemma item 1). C_2^+ intersects C_1^+, thus they are all in the same clique C^∞ (item 2), which is the union of C_1^+ and C_2^+ (item 3). A chain of intersecting cliques connects C_1^+ to C_2^+. Item 4 follows quite directly from item 3; in Figure 11, it is illustrated e.g., for properties a and c by taking $D_1 = C_1$, $D_2 = C_2$.

Based on Theorem 2, we show:

Theorem 3 ($\mathtt{W}$ shortcut)**.** *The shortcut holds for* $\equiv_{\mathtt{W}}$.

For instance, on the graph in Figure 9, it is easy to check that applying summarization on $(\mathtt{G}_{/\mathtt{W}})^\infty$ (as prescribed by the shortcut) leads exactly to a graph strongly isomorphic to $(\mathtt{G}^\infty)_{/\mathtt{W}}$.

Showing Theorem 3 is rather involved. We do it in several steps. First, based on Lemma 1, we show:

Lemma 2 (Property relatedness in $\mathtt{W}$ summaries)**.** *Data properties are target-related (resp. source-related) in* $(\mathtt{G}_{/\mathtt{W}})^\infty$ *iff they are target-related (resp. source-related) in* $\mathtt{G}^\infty$.

Based on the above and on Theorem 1, we establish the next result from which Theorem 3 directly follows:

Proposition 4. (SAME CLIQUES-$\mathtt{W}$) $\mathtt{G}^\infty$ *and* $(\mathtt{G}_{/\mathtt{W}})^\infty$ *have identical source clique sets, and identical target cliques sets. Further, a node* $n \in \mathtt{G}^\infty$ *has exactly the same source and target clique as* $f_{\mathtt{W}}(n)$ *in* $(\mathtt{G}_{/\mathtt{W}})^\infty$.

Theorem 4 ($\mathtt{S}$ shortcut)**.** *The shortcut holds for* $\equiv_{\mathtt{S}}$.

We prove this based on counterparts of statements established for $\mathtt{G}_{\mathtt{W}}$. First we show:

Lemma 3 (Property relatedness in $\mathtt{S}$ summaries)**.** *Data properties are target-related (resp. source-related) in* $(\mathtt{G}_{/\mathtt{S}})^\infty$ *iff they are target-related (resp. source-related) in* $\mathtt{G}^\infty$.

Then, from Theorem 1 and the above Lemma, we obtain the next proposition from which Theorem 4 directly follows:

Proposition 5. (SAME CLIQUES-$\mathtt{S}$) $\mathtt{G}^\infty$ *and* $(\mathtt{G}_{/\mathtt{S}})^\infty$ *have identical source clique sets, and identical target clique sets. Further, a node* $n \in (\mathtt{G}_{/\mathtt{S}})^\infty$ *has exactly the same source and target clique as* $f_{\mathtt{S}}(n)$ *in* $(\mathtt{G}_{/\mathtt{S}})^\infty$.

Finally, we have:

Theorem 5 (No shortcut for $\equiv_{\mathtt{TW}}$)**.** *The shortcut does not hold for* $\equiv_{\mathtt{TW}}$.

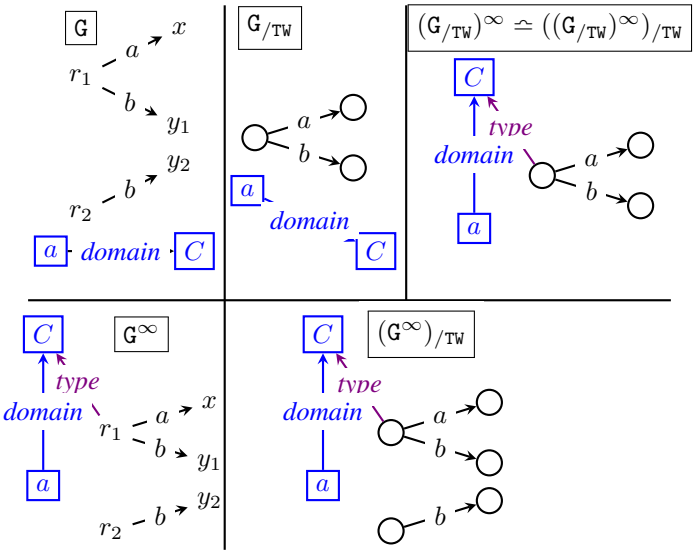


Figure 12: Shortcut counter-example.

We prove this by exhibiting in Figure 12 a counter-example. In $\mathtt{G}$ and $\mathtt{G}_{/\mathtt{TW}}$, all data nodes are untyped; only after saturation a node gains the type C. Thus, in $\mathtt{G}_{/\mathtt{TW}}$, one (untyped) node represents all data property subjects; this is exactly a "hasty fusion" as discussed below Theorem 2. In $(\mathtt{G}_{/\mathtt{TW}})^\infty$, this node gains a type, and in $((\mathtt{G}_{/\mathtt{TW}})^\infty)_{/\mathtt{TW}}$, it is represented by a single node. In contrast, in $\mathtt{G}^\infty$, r_1 is typed and r_2 isn't, leading to two distinct nodes in $(\mathtt{G}^\infty)_{/\mathtt{TW}}$. This is not isomorphic with $(\mathtt{G}_{/\mathtt{TW}})^\infty$ which, in this example, is strongly isomorphic to $((\mathtt{G}_{/\mathtt{TW}})^\infty)_{/\mathtt{TW}}$. Thus, the shortcut does not hold for $\equiv_{/\mathtt{TW}}$ does not admit a shortcut.

Theorem 6 (No shortcut for $\equiv_{\mathtt{TS}}$)**.** *The shortcut does not hold for* $\equiv_{\mathtt{TS}}$.

The graph in Figure 12 is also a shortcut counter-example for $\mathtt{TS}$. More generally, let $\equiv_{\mathtt{X}}$ be an arbitrary RDF node equivalence, and $\equiv_{\mathtt{TX}}$ be a type-first summary obtained by replacing in Definition 7, $\equiv_{\mathtt{W}}$ by $\equiv_{\mathtt{X}}$. Based on this counter-example, one can show that the shortcut does not hold for $\equiv_{\mathtt{TX}}$. If the ontology only features subclass triples, the shortcut holds also for $\equiv_{\mathtt{TW}}$ and $\equiv_{\mathtt{TS}}$; this is because any node typed in $\mathtt{G}^\infty$ was already typed in $\mathtt{G}$.

Based on Theorem 2, we have also established:

Theorem 7. (BISIMILARITY SHORTCUT) *The shortcut holds for the forward ($\equiv_{\mathtt{fb}}$), backward ($\equiv_{\mathtt{bw}}$), and forward-and-backward ($\equiv_{\mathtt{fb}}$) bisimilarity equivalence relations (recalled in Section 3).*

6.4 Relationships between summaries

On any graph $\mathtt{G}$, it follows from our node equivalence relations that any two nodes that are $\equiv_{\mathtt{S}}$ are also $\equiv_{\mathtt{W}}$. Based on this, it is easy to show that $(\mathtt{G}_{/\mathtt{S}})_{/\mathtt{W}} = \mathtt{G}_{/\mathtt{W}}$, i.e., one could compute $\mathtt{G}_{/\mathtt{W}}$ by first summarizing $\mathtt{G}$ into $\mathtt{G}_{/\mathtt{S}}$, and then applying weak summarization on this (typically much smaller) graph; similarly, $(\mathtt{G}_{/\mathtt{TS}})_{/\mathtt{TW}} = \mathtt{G}_{/\mathtt{TW}}$. It is also the case that $(\mathtt{G}_{/\mathtt{W}})_{/\mathtt{S}} = \mathtt{G}_{/\mathtt{W}}$, i.e., strong summarization cannot compress a weak summary further, and similarly $(\mathtt{G}_{/\mathtt{TW}})_{/\mathtt{TS}} = \mathtt{G}_{/\mathtt{TW}}$. Figure 13 summarizes the main relationships between $\mathtt{G}$, $\mathtt{G}^\infty$, our summaries and bisimilarity-based ones.

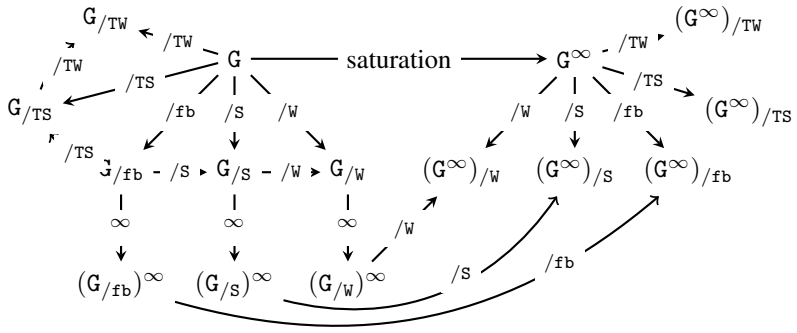


Figure 13: Relations between quotient summaries.

7 Summarization algorithms

We now discuss summarization algorithms which, given as input a graph $\mathtt{G}$, construct $\mathtt{G}_{/\mathtt{W}}$, $\mathtt{G}_{/\mathtt{S}}$, $\mathtt{G}_{/\mathtt{TW}}$ and $\mathtt{G}_{/\mathtt{TS}}$. The schema $\mathtt{S}_\mathtt{G}$ is directly copied in the summary thus, below we focus on summarizing $\mathtt{D}_\mathtt{G}$ (Section 7.1) and $\mathtt{T}_\mathtt{G}$ (Section 7.2).

7.1 Data graph summarization

We have devised summarization algorithms of two flavors.

(A) **Global** algorithms start by learning the equivalence relation[6] and creating the summary nodes. Then, a final pass over $\mathtt{G}$ computes $f_\equiv$ and adds to the summary the edge $f_\equiv(n_1)\ e\ f_\equiv(n_2)$ for each triple $n_1\ e\ n_2$ in $\mathtt{G}$. While this requires several passes over the graph, it connects its nodes directly to their final representatives they will have in $\mathtt{G}_{/\equiv}$.

We start with our **global** $\mathtt{W}$ **summarization algorithm** (Table 3). It exploits Proposition 1, which guarantees that any data property occurs only once in the summary. To each distinct data property $\mathtt{p}$ encountered in $\mathtt{D}_\mathtt{G}$, it associates a summary node (integer) $s_\mathtt{p}$ which will be the (unique) source of $\mathtt{p}$ in the summary, and similarly a node $t_\mathtt{p}$ target of $\mathtt{p}$; these are initially unknown, and evolve as $\mathtt{G}$ is traversed. Further, it uses two maps op and ip which associate to each $\mathtt{D}_\mathtt{G}$ node n, the set of its outgoing, resp. incoming data properties. These are filled during the first traversal of $\mathtt{D}_\mathtt{G}$ (step 1.) Steps 2. to 2.5 ensure that for each node n having outgoing properties and possibly incoming ones, $s_\mathtt{p}$ for all the outgoing ones are equal, and equal also to $t_\mathtt{p}$ for all the incoming ones. This is performed using a function $fuse$ which, given a set of summary nodes, picks one that will replace all of them. In our implementation, summary nodes are integers, and $fuse$ is simply min; we just need $fuse$ to be distributive over $\cup$, i.e., $fuse(A, (B \cup C)) = fuse(fuse(A, B), fuse(A, C))$. Step 3. symetrizes this to ensure that the incoming properties of nodes lacking outgoing properties (thus, absent from op) also have the same target. In Step 4., we represent $\mathtt{s}$ and $\mathtt{o}$ based on the source/target of the property $\mathtt{p}$ connecting them. The $fuse$ operations in 2. and 3. have ensured that, while traversing $\mathtt{G}$ triples in 4., a same $\mathtt{G}$ node n is always represented by the same summary node $f_\mathtt{W}(n)$.

Our **global** $\mathtt{S}$ **summarization algorithm** (Table 4) uses two maps sc and tc which store for each data node $n \in \mathtt{D}_\mathtt{G}$, its source clique $sc(n)$, and its target clique $tc(n)$, and for each data property $\mathtt{p}$, its source clique $src_\mathtt{p}$ and target clique $trg_\mathtt{p}$. Further, to each (source clique, target clique) pair encountered until

[6] Recall that $\equiv_\mathtt{W}$, $\equiv_\mathtt{S}$, $\equiv_\mathtt{T}$, as well as bisimilarity equivalence, are defined *based on the data/type triples of a given graph* $\mathtt{G}$, thus when starting to summarize $\mathtt{G}$, we do not know whether any two nodes are equivalent; the full $\equiv$ is known only after inspecting all $\mathtt{G}$ triples.

Algorithm **global-W(G)**

1. For each $\mathtt{s}\ \mathtt{p}\ \mathtt{o} \in \mathtt{G}$, add $\mathtt{p}$ to $op(\mathtt{s})$ and to $ip(\mathtt{o})$.
2. For each node $n \in op$:
2.1. Let $X \leftarrow fuse\{s_\mathtt{p} \mid \mathtt{p} \in op(n)\}$.
If X is undefined, let $X \leftarrow$nextNode();
2.2. Let $Y \leftarrow fuse\{t_\mathtt{p} \mid \mathtt{p} \in ip(n)\}$.
If Y is undefined, let $Y \leftarrow$nextNode();
2.3. Let $Z \leftarrow fuse(X, Y)$;
2.4. For each $\mathtt{p} \in ip(n)$, let $s_\mathtt{p} \leftarrow Z$;
2.5. For each $\mathtt{p} \in op(n)$, let $t_\mathtt{p} \leftarrow Z$;
3. Repeat **2** to **2.5** swapping ip with op and $t_\mathtt{p}$ with $s_\mathtt{p}$;
4. For each $\mathtt{s}\ \mathtt{p}\ \mathtt{o} \in \mathtt{G}$: let $f_\mathtt{W}(\mathtt{s}) \leftarrow s_\mathtt{p}$, $f_\mathtt{W}(\mathtt{o}) \leftarrow t_\mathtt{p}$;
Add $f_\mathtt{W}(\mathtt{s})\ \mathtt{p}\ f_\mathtt{W}(\mathtt{o})$ to $\mathtt{G}_{/\mathtt{W}}$.

Table 3: Global W summarization algorithm.

Algorithm **global-S(G)**

1. For each $\mathtt{s}\ \mathtt{p}\ \mathtt{o} \in \mathtt{G}$:
1.1. Check if $src_\mathtt{p}$, $trg_\mathtt{p}$, $sc(\mathtt{s})$ and $tc(\mathtt{o})$ are known; those not known are initialized with $\{\mathtt{p}\}$;
1.2. If $sc(\mathtt{s}) \neq src_\mathtt{p}$, fuse them into new clique $src'_\mathtt{p} = sc(\mathtt{s}) \cup src_\mathtt{p}$; similarly, if $tc(\mathtt{o}) \neq trg_\mathtt{p}$, fuse them into $trg'_\mathtt{p} = tc(\mathtt{o}) \cup trg_\mathtt{p}$. **2.** For each $\mathtt{s}\ \mathtt{p}\ \mathtt{o} \in \mathtt{G}$:
2.1. $f_\mathtt{S}(\mathtt{s}) \leftarrow$ the (unique) summary node corresponding to the cliques $(sc(\mathtt{s}), tc(\mathtt{s}))$; similarly, $f_\mathtt{S}(\mathtt{o}) \leftarrow$ the node corresponding to $(sc(\mathtt{o}), tc(\mathtt{o}))$ (create the nodes if needed).
2.2 Add $f_\mathtt{S}(\mathtt{s})\ \mathtt{p}\ f_\mathtt{S}(\mathtt{o})$ to $\mathtt{G}_{/\mathtt{S}}$.

Table 4: Global S summarization algorithm.

a certain point during summarization, we store the (unique) corresponding summary node. Steps 1.-1.2. build the source and property cliques present in G and associate them to every subject and object node (in sc and tc), as well as to any data property (in $src_\mathtt{p}$ and $trg_\mathtt{p}$). For instance, on the sample graph in Figure 2, these steps build the cliques in Table 2. Steps 2-2.2. represent the nodes and edges of G.

The correctness of algorithms **global-W** and **global-S** follows quite easily from their descriptions and the summary definitions.

(B) **Incremental** algorithms simultaneously learn the equivalence relation from G *and* represent G data triples. They are particularly suited for incremental **summary maintenance**: if new triples $\Delta^+_\mathtt{G}$ are added to G, it suffices to run the summarization algorithm only on $\Delta^+_\mathtt{G}$, based on $\mathtt{G}_\equiv$ and its representation function $f_\equiv$, in order to obtain $(\mathtt{G} \cup \Delta^+_\mathtt{G})_{/\equiv}$. Incremental algorithms also provide the basic building blocks for incrementally propagating the effect of a deletion from G. However, incremental algorithms are considerably **more complex**, since various decisions (assigning sources/targets to properties in W, source/target cliques in S, node representatives in both) must be *repeatedly revisited* to reflect newly acquired knowledge. We illustrate this on our algorithms and examples below.

Each **incremental summarization** algorithm consists of an **incremental update method, called for every $\mathtt{D}_\mathtt{G}$ triple**, which adjusts the summary's data structures, so that at any point, the summary reflects exactly the graph edges (triples) visited until then.

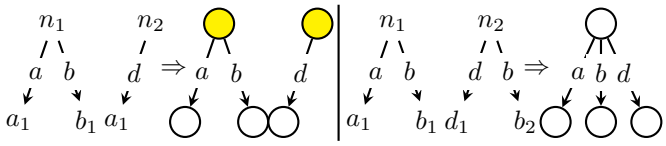
Table 5 outlines incremental W summarization. For example (see the figure below), assume the algorithm traverses the graph G in Figure 2 starting with: $n_1\ a\ a_1$, then $n_1\ b\ b_1$, then $n_2\ d\ d_1$. When we summarize this third triple, we do not know yet that the source of a d triple is also equivalent to n_1, be-

Algorithm **increm-W(s p o)**

1. Check if $s_\mathtt{p}$ and $o_\mathtt{p}$ are known: either both are known (if a triple with property p has already been traversed), or none;
2. Check if $f_\mathtt{W}(\mathtt{s})$ and $f_\mathtt{W}(\mathtt{o})$ are known; none, one, or both may be, depending on whether s, respectively o have been previously encountered;
3. Fuse $s_\mathtt{p}$ with $f_\mathtt{W}(\mathtt{s})$ (if one is unknown, assign it the value of the other), and $o_\mathtt{p}$ with $f_\mathtt{W}(\mathtt{o})$;
4. Update $f_\mathtt{W}(\mathtt{s})$ and $f_\mathtt{W}(\mathtt{o})$, if needed;
5. Add the edge $f_\mathtt{W}(\mathtt{s})$ p $f_\mathtt{W}(\mathtt{o})$ to $\mathtt{G}_{/\mathtt{W}}$.

Table 5: Incremental W summarization of one triple.

cause no common source of b and d (e.g., n_2 or n_4) has been seen so far. Thus, n_2 is found not equivalent to any node visited so far, and represented separately from n_1. Now assume the fourth triple traversed is $n_2\ b\ b_2$: at this point, we know that a, b and d are in the same source clique, thus $n_1 \equiv_\mathtt{W} n_2$, and their representatives (highlighted in yellow below) must be **fused** in the summary (Step 3.) More generally, it can be shown that $\equiv_\mathtt{W}$ **only grows** as more triples are visited, in other words: if in a subset $\mathtt{G}'$ of G's triples, two nodes n_1, n_2 are weakly equivalent, then this holds in any $\mathtt{G}''$ with $\mathtt{G}' \subseteq \mathtt{G}'' \subseteq \mathtt{G}$.



Summary node **fusion** dominates the algorithm's **complexity**. Let N_1, N_2 be two sets of G nodes, represented at a certain point by the distinct summary nodes m_1, m_2. When fusing the latter into a single m, we must also record that all the nodes in $N_1 \cup N_2$ are now represented by m. A naïve implementation leads to $O(N^2)$ complexity, where N is the number of nodes in $\mathtt{D}_\mathtt{G}$, since each new node may lead to a fusion whose cost is $O(N)$; in the worst case N could be proportional to $|\mathtt{G}|$, the number of triples in G, leading to an overall complexity of $O(|\mathtt{G}|^2)$ for the incremental weak summarization.

Instead, we rely on a **Union-Find** (aka Disjoint Sets) data structure, with the *path compression* and *union by size*[7] optimizations, which guarantee an overall **quasi-linear worst-case complexity** to our incremental weak summarization algorithm. The exact complexity is $O(N\alpha(N))$ where $\alpha(N)$, the inverse Ackermann's function, is smaller than 5 for any machine-representable input N. Assimilating this to **linear-time**, the algorithm's complexity class is in $O(|\mathtt{G}|)$, which is also **optimal**, as summarization cannot do less than fully traversing G.

Table 6 outlines the incremental update of the S summary due to the traversal of the triple s p o. Conceptually, the algorithm is symetric for the source (s) and target (o) of the edge, we only discuss the source side below. Steps 1. and 2. start by determining the source clique of s, based on its previously known source clique (if any) and the previously known source clique of p (if any); after step 2., s's source (and target) clique reflecting also the newly seen triple s p o are completely known. Determining them may have involved fusing some previously separate cliques. For instance, on the graph in Figure 2, assume we first traverse the two a triples, then we traverse $n_2\ b\ b_2$; so far we have the source cliques $\{a\}$, $\{b\}$ and $\emptyset$. If the next traversed triple is $n_2\ a\ a_2$, we fuse the source cliques (step 3.1) $\{a\}$ and $\{b\}$ into $\{a, b\}$. This requires fusing the summary node whose (source, target) cliques were ($\{a\}$, $\emptyset$) with the one which had ($\{b\}$, $\emptyset$) (Step 3.2).

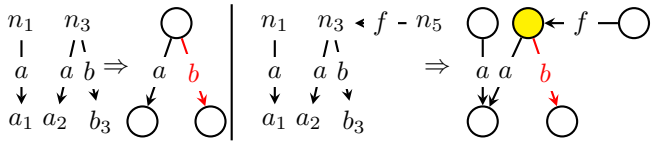
The last intricacy of incremental strong summarization is due to the fact that unlike $\equiv_\mathtt{W}$, $\equiv_\mathtt{S}$ **may grow and shrink** during incremental, strong summarization. For instance, assume incremental strong

[7] https://en.wikipedia.org/wiki/Disjoint-set_data_structure

Algorithm **increm-S(s p o)**
1. Check if we already know a source clique $src_{\mathtt{p}}$ (resp. target clique $trg_{\mathtt{p}}$). Either both are known (if a p triple has already been traversed), or none. Those not known are initialized with $\{\mathtt{p}\}$;
2. Check if $sc(\mathtt{s})$ (resp. $tc(\mathtt{o})$) are known; those unknown are initialized with $\{\mathtt{p}\}$;
3. If $sc(\mathtt{s}) \neq src_{\mathtt{p}}$, fuse them into new clique $src'_{\mathtt{p}} = sc(\mathtt{s}) \cup src_{\mathtt{p}}$, using Union-Find; similarly, if $tc(\mathtt{o}) \neq trg_{\mathtt{p}}$, fuse them into $trg'_{\mathtt{p}} = tc(\mathtt{o}) \cup trg_{\mathtt{p}}$, and:
3.1 Replace $sc(\mathtt{s})$ and $src_{\mathtt{p}}$ with $src'_{\mathtt{p}}$ throughout the summary (respectively, replace $tc(\mathtt{o})$ and $trg_{\mathtt{p}}$ with $trg'_{\mathtt{p}}$);
3.2 The above may entail summary node fusions; in this case, update $f_{\mathtt{S}}$ (use Union-Find) and the summary edges to reflect it;
4. If before seeing s p o s had been already represented and it had an empty source clique, then s needs to split, i.e., be represented separately from the nodes to which it was $\equiv_{\mathtt{S}}$ previously; call **split-source(s)**. (Symetric discussion for o, call **split-target(o)**).
5. Update $f_{\mathtt{S}}(\mathtt{s})$ and $f_{\mathtt{S}}(\mathtt{o})$, if needed;
6. Add the edge $f_{\mathtt{S}}(\mathtt{s})$ p $f_{\mathtt{S}}(\mathtt{o})$ to $\mathtt{G}_{/\mathtt{S}}$.

Table 6: Incremental S summarization of one triple.

summarization of the graph in Figure 2 starts with $n_1\ a\ a_1$, $n_3\ a\ a_2$, $n_3\ b\ b_3$ (see the figure below). After these, we know $n_1 \equiv_{\mathtt{S}} n_3$; their source clique is $\{a, b\}$ and their target clique is $\emptyset$. Assume the next triple traversed is $n_5\ f\ n_3$: at this point, n_3 is *not* $\equiv_{\mathtt{S}}$ *to* n_1 *any more*, because n_5's target clique is now $\{f\}$ instead of the empty $\emptyset$. Thus, n_5 **splits** from n_1, that is, it needs to be represented by a new summary node (shown in yellow below), distinct from the representative of n_1.



Further, note that the representative of n_1 and n_3 (at left above) had one b edge (highlighted in red) which was *solely due to* n_3*'s outgoing* b *edge*. By definition of a quotient summary (Section 3), that edge *moves* from the old to the new representative of n_3 (the yellow node). If, above at left, n_1 had also had an outgoing edge labeled b, at right, both nodes in the top row would have had an outgoing b edge. It can be shown that *splits only occur in such cases*, i.e., o whose target clique becomes non-empty (respectively, s whose source clique becomes non-empty, and the node was previously represented together with other nodes; if it was represented alone, we just update the respective clique of its representative).

The procedure **split-source(s)** (ommitted for space reasons) represents s separately, to reflect it no longer has an empty target clique, and, for each outgoing edge of s: adds a corresponding edge to the new representative of s; and checks if, as a consequence, an edge needs to be removed from its previous representative.

Proposition 6. (ALGORITHM CORRECTNESS) *Applying algorithm* **increm-W** *(respectively,* **increm-S***) successively on each triple of* G*, in any order, builds* $\mathtt{G}_{/\mathtt{W}}$ *(respectively,* $\mathtt{G}_{/\mathtt{S}}$*).*

Splitting requires inspecting the data edges attached to the splitting node, in order to add to its new representative the edges it must have (such as the $n_3\ b\ b_3$ above). We make the hypothesis, denoted (★), that the maximum number of edges incoming/outgoing a data node is small (and basically constant) compared to the size of G; this was the case in the graphs we have experimented with. To keep splitting cost under control: (i) We store for each summary node m and edge e a *counter* $m_\sharp, e_\sharp$ of the $\mathtt{D}_{\mathtt{G}}$ nodes

and edges it represents. Splitting is only needed when $m_\sharp > 1$. (ii) Summary node m loses an outgoing (resp. incoming) edge e labeled p when s (resp. o) splits, if and only if the number of outgoing s edges (resp. incoming o edges) labeled p equals $e_\sharp$. At left in the figure above, $e_\sharp$ was 1, thus the b edge is removed from the old representative of n_3.

Under the (★) hypothesis, using the data structures (including Union-Find) described above, **the complexity of incremental strong summarization is amortised constant per added triple**.

All our algorithms require $O(|\mathtt{G}|)$ space to store the summary edges, the representation function, and their other data structures.

7.2 Typed graph summarization

We now explain how to extend our incremental $\mathtt{D}_\mathtt{G}$ summarization algorithms to type triples.

To extend W, respectively, S summarization to type triples in "data-then-type" fashion (Section 5.1), we run W, resp. S summarization *first, over* $\mathtt{D}_\mathtt{G}$ *triples only*, as described in Section 7.1. This assigns their (final) representatives to all nodes of $\mathtt{D}_\mathtt{G}$. Then, for each s *type* C triple, we simply add to the summary the edge $f_\mathtt{W}(\mathtt{s})$ *type* C (resp. $f_\mathtt{S}(\mathtt{s})$ *type* C); recall from Section 5 that any class node C is represented by itself.

For "type-then-data" summarization (Section 5.2), that is, for TW and TS, *we first traverse* $\mathtt{T}_\mathtt{G}$ *triples only*, compute all the class sets, and assign to each typed data node a representative based on its class set. Then, we run a *type-aware variant* of a W (resp. S) algorithm, either global or incremental. The changes introduced in the type-aware variant are: (i) In TW summarization, a data property p may lack an untyped source, if p only occurs on typed nodes; consider for instance the graph whose only triples are n_1 *type* C, n_1 e a_1. Similarly, in TS summarization, a property (e.g., e above) may have a target clique, but lack a source clique, since it does not have an untyped source. (ii) Summarizing the data triple s p o does not fuse nor split the representative of s (resp. o) if s (resp. $object$) is typed; instead, representatives of typed nodes (computed during the *type* triples traversal) are left untouched.

Proposition 7. (ALGORITM CORRECTNESS) *Applying global-*W *(respectively global-*S*) on* G*, or applying increm-*W *(respectively, increm-*S*) on each triple of* G*, extended as described above for data-then-type or type-then-data summarization leads, respectively* $\mathtt{G}_{/\mathtt{W}}$, $\mathtt{G}_{/\mathtt{S}}$, $\mathtt{G}_{/\mathtt{TW}}$ *and* $\mathtt{G}_{/\mathtt{TS}}$.

These algorithms need to work with $\mathtt{S}_\mathtt{G}$, $\mathtt{D}_\mathtt{G}$ and $\mathtt{T}_\mathtt{G}$ *separately*. Fortunately, most popular RDF store allow such direct access. The space needed to also represent $\mathtt{T}_\mathtt{G}$ triples remains linear in $|\mathtt{G}|$.

8 Experimental study

Algorithms and settings. We have implemented our algorithms as well as fw, bw, fb and 1fb summarization in Java 1.8. We used the recent [25] algorithm for fw, bw and fb and devised our simple algorithm for 1fb. We deployed them using PostgreSQL v9.6 to store RDF triples in an integer-encoded triple table, indexed by s, p and o; the server had 30 GB of shared buffers and 640 MB working memory. The JVM had 90 GB of RAM. We used a Linux server with an Intel Xeon CPU E5-2640 v4 @2.40GHz and 124 GB RAM.

Datasets. We have experimented with numerous real and synthetic datasets, from 10^5 to $1.4 \cdot 10^8$ triples; the largest file is 36.5 GB on disk. Table 7 shows for each graph its number of triples $|\mathtt{G}|$, the number of triples in the saturated graph $|\mathtt{G}^\infty|$, the number of schema triples $|\mathtt{S}_\mathtt{G}|$, and the number of distinct data properties #p and classes #C.

Compression factors. Figure 14 shows the ratio $|\mathtt{G}|/|\mathtt{G}_{/\equiv}|$, called the *compression factor* $cf_\equiv$ for our graphs. To our summaries we added 1fb, fw, bw and fb; we plot fw and fb, as bw was somewhere in-between. Some fw and bw summarizations ran out of memory or were stopped after more than 2 hours.

Real datasets	$\|\mathtt{G}\|$	$\|\mathtt{G}^{\infty}\|$	$\|\mathtt{S}_{\mathtt{G}}\|$	$\#\mathtt{p}$	$\#\mathtt{C}$
DBLP	88,153,334	88,153,334	0	26	14
DBpedia Person	7,889,268	7,889,268	0	9	1
INSEE Geo	369,542	1,112,803	196	53	144
Springer LOD	145,136	213,017	40	26	16
Nobel Prizes	80,757	109,901	35	45	28
Synth. datasets	$\|\mathtt{G}\|$	$\|\mathtt{G}^{\infty}\|$	$\|\mathtt{S}_{\mathtt{G}}\|$	$\#\mathtt{p}$	$\#\mathtt{C}$
LUBM [16] 1M	1,227,744	1,227,744	0	17	17
LUBM 10M	11,990,059	11,990,059	0	17	21
LUBM 100M	114,355,295	114,355,295	0	17	21
BSBM [2] 1M	1,000,708	1,009,138	150	38	159
BSBM 10M	10,538,484	10,628,484	584	38	593
BSBM 100M	104,115,556	105,315,556	2,010	38	2019
BSBM 138M	138,742,476	140,342,476	2,412	38	2,421

Table 7: Datasets used in experiments.

For W and S, cf is close to or above 10^3, and reaches $3 \cdot 10^6$ in some cases; in all our experiments, $cf_{\mathtt{W}}$ was the highest. In contrast, $cf_{\mathtt{fb}}$ rounds to 1 up to 3, since full bisimilarity is rare. Since $\equiv_{\mathtt{fw}}$ is a weaker condition and $\equiv_{\mathtt{1fb}}$ even weaker, $cf_{\mathtt{fw}}$ and especially $cf_{\mathtt{1fb}}$ are higher, but still up to 5 times lower than $cf_{\mathtt{W}}$. *In all our experiments, the weak summary is the most compact;* S*,* TW *and* TS *are close, followed by* 1fb*, from* 2.9 *to* 6.9 *times larger*. Conversely, full bisimulation achieves little to no compression, while fw (which has the drawback of being asymetric) compresses less than 1fb.

Summarization time. The time to build $\mathtt{G}_{/\mathtt{W}}$, $\mathtt{G}_{/\mathtt{S}}$, $\mathtt{G}_{/\mathtt{TW}}$ and $\mathtt{G}_{/\mathtt{TS}}$ using the global and the incremental algorithms, as well as the time to build $\mathtt{G}_{/\mathtt{1fb}}$, are plotted as a function of $|\mathtt{G}|$ in Figure 15; both axes are in log scale. For each summary type, the summarization time is roughly linear in the size of G, confirming the expectations stated in Section 7; they range from 200 ms to 34.5 minutes (increm-S on the BSBM138M). Increm-W is the fastest overall; it traverses G only once, and it has relatively little processing to do, thus it is faster than global-W which performs several passes, but does not need to replace node representatives. 1fb summarization time is close, then we see the global S, TW and TS, and finally incremental S which, as we explained, is quite complex. The fact that increm-W is the cheapest and increm-S the most expensive show that the former may be I/O-bound, while the latter (with the same I/O cost) is CPU (and memory)-bound. Since increm-S is rather expensive per-triple, it is more efficient to first summarize a graph using global-S, and call increm-S only to maintain it later. This is significantly faster: for instance, global-S on BSBM138M takes only 11.85 minutes. Increm-TS is often faster than increm-S because typed nodes do not lead to splits during TS summarization.

Shortcut speed-up. Table 8 shows the time to build $(\mathtt{G}^{\infty})_{/\equiv}$ in two ways: (i) *direct*, i.e., saturate G then summarize, denoted $dt_{\equiv}$, and (ii) *shortcut*, summarize G, then saturate the summary and summarize again, denoted $st_{\equiv}$. We define the **shortcut speed-up** $x_{\equiv}$ for $\equiv \in \{\mathtt{W}, \mathtt{S}\}$ as $(dt_{\equiv} - st_{\equiv})/dt_{\equiv}$ and report it also in the table. The speed-up is highest for $\mathtt{G}_{/\mathtt{W}}$ (up to almost 98%) and $\mathtt{G}_{/\mathtt{S}}$ (up to 95%): this is a direct consequence of their good compression. Indeed, $dt_{\equiv}$ includes the time to summarize $\mathtt{G}^{\infty}$, while $st_{\equiv}$ includes the time to summarize $(\mathtt{G}_{/\equiv})^{\infty}$; the smaller this is, the higher $x_{\equiv}$. The table confirms the practical interest of the shortcut (Section 6) for summarizing the full semantics of a graph G.

Experiment conclusion. Our experiments have shown that our four summaries can be built efficiently, and they reduce graph sizes by factors hundreds up to $3 \cdot 10^6$; they are *two to three orders of magnitude more compact than summaries based on full bisimulation*; $\mathtt{G}_{/\mathtt{W}}$ is the most compact, and the other three

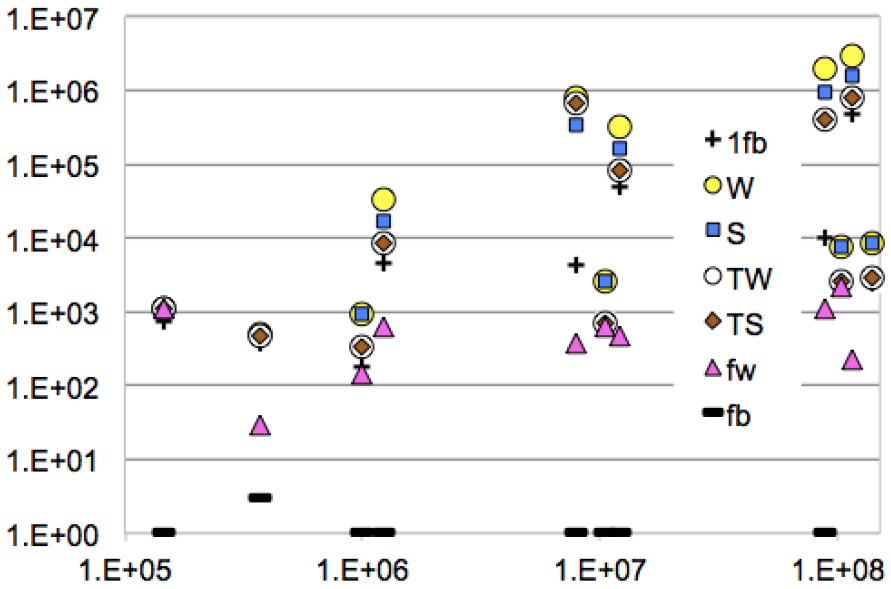


Figure 14: Summarization compression factors.

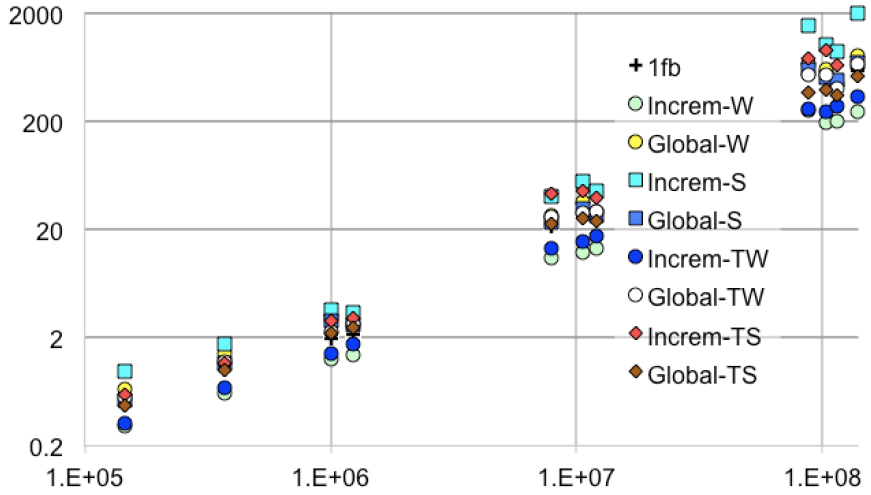


Figure 15: Summarization time (s) for various graph sizes $|\mathtt{G}|$.

summaries we introduced are close (within a factor of 3). Finally, among the summaries which admit a shortcut ($\mathtt{G}_{/\mathtt{W}}$, $\mathtt{G}_{/\mathtt{S}}$, $\mathtt{G}_{/\mathtt{fb}}$ and $\mathtt{G}_{/\mathtt{fw}}$), *the shortcut speed-up is up to 98% for* $\mathtt{G}_{/\mathtt{W}}$*, and 95% by* $\mathtt{G}_{/\mathtt{S}}$. All our algorithms scale linearly with $|\mathtt{G}|$; increm-W is the fastest, while increm-S is the slowest. Overall, if summary size or building time are a critical concern, we recommend using $\mathtt{G}_{/\mathtt{W}}$; otherwise, $\mathtt{G}_{/\mathtt{S}}$ gives finer-granularity information. TW and TS summaries should be used when data types are deemed most important in order to understand the data. However, to summarize $\mathtt{G}^{\infty}$, only the direct path (saturate, then

Dataset	$dt_{\mathtt{W}}$ (s)	$st_{\mathtt{W}}$ (s)	$x_{\mathtt{W}}$ (%)	$dt_{\mathtt{S}}$ (s)	$st_{\mathtt{S}}$ (s)	$x_{\mathtt{S}}$ (%)
INSEE Geo	35.85	0.81	97.73	38.59	1.95	94.96
Springer	3.96	0.45	88.60	4.59	1.159	74.73
Nobel	2.13	0.41	80.56	2.60	0.75	71.09
BSBM1M	5.45	1.48	72.85	9.35	3.82	59.20
BSBM10M	71.63	12.67	82.32	142.46	55.64	60.94
BSBM100M	989.00	198.00	79.98	1715.40	1030.36	39.93
BSBM138M	1393.69	251.93	81.92	3627.19	2049.22	43.50

Table 8: Shortcut experiments.

summarize) is available for these.

9 Related Work

Graph summarization has a long history and many applications; a recent survey is [28]. It may rely on *graph structure*, *graph values* or *statistics*, e.g., how many distinct values a property has, how many edges are adjacent to a node etc. Summaries may reflect (albeit with some loss of information) the *complete* graph, or they may focus just on *part* of it, e.g., the highest-ranked nodes according to a metric such as PageRank, those having changed recently etc.

In this work, we study *structural quotient summaries*, which are *complete* and *representative* as discussed in Section 3. Our first core contribution is defining property cliques, which lead to the novel notions of weak and strong equivalence, in turn leading to novel quotient summaries for any labeled graph and in particular RDF. Quotient summaries most widely studied in the literature are based on *bisimulation* [29, 9, 21, 12]; they can be built in $O(|\mathtt{G}| \cdot \log(N))$, where N is the number of nodes in $\mathtt{G}$. Forward-and-backward bisimulation $\equiv_{\mathtt{fb}}$, symmetric w.r.t. edge directions, is most suited to RDF. However, it is well known, e.g., shown in [25], that heterogeneous graph nodes are very rarely $\equiv_{\mathtt{fb}}$, thus $\equiv_{\mathtt{fb}}$ summaries barely compress $\mathtt{G}$ (recall Figure 14). $\equiv_{\mathtt{1fb}}$ is more permissive, and can be seen as the closest competitor of $\equiv_{\mathtt{W}}$ and $\equiv_{\mathtt{S}}$; we have shown $\equiv_{\mathtt{1fb}}$ still leads to summaries several times larger than ours. We view our property clique-based summaries as **complementary to** bisimulation-based ones: ours cope better with the heterogeneity of RDF graphs, thus are more suited for visualisation (they were precisely chosen for that in LODAtlas [32]), while bisimulation-based ones lead to larger summaries but allow, e.g., finding "all nodes having properties a and b" as those represented by a $\mathtt{G}_{/\mathtt{1fb}}$ node having an a and a b. If one of our summaries is used for this task, a *superset* of the desired nodes is obtained. Bisimulation- and clique-based summaries each have distinct advantages, and can be used side-by-side for different purposes.

The well-known Dataguides [15] are not quotients, as a graph node may be represented by *more than one* node. A Dataguide may be larger than the original graph, and its construction has worst-case exponential time complexity in the size of $\mathtt{G}$. [13] considers *reachability* and *graph pattern* queries on labeled graphs, and builds *answer-preserving summaries* for such queries. However, their query semantics is *not based on graph homomorphisms*, which underlie standard (and SPARQL) query semantics, but on bounded graph simulation. Under this semantics, answering a query becomes P (instead of NP), at the price of not preserving the query structure (i.e., joins), which quotient summaries do preserve.

With a focus farther from our work, [8] introduces an *aggregation* framework for OLAP on labeled graphs, while we focus on representing complete graph structure and semantics. [7] builds a set of randomized summaries to be *mined* instead of the original graph for better performance, with guaranteed bounds on the information loss. Graph compression with bounded "error" (number of edges to be added as "corrections" after decompression, to retrieve the original graph) is studied in [30, 23]. Quantitative

summarization, where nodes and edges are summarized by their counts, is the focus of [26]. [27] surveys many other quantitative, mining-oriented graph sampling and summarization methods.

Focusing on RDF graphs, [34, 36, 10] study bisimulation-based RDF quotient summaries, providing efficient parallel summarization algorithms [10] and showing they are representative [34]. However, these summaries ignore RDF saturation, and thus its interaction with summarization. Summaries based on clustering [17], user-defined aggregation rules [33], mining [7], and identification of frequent subtrees [38] are not complete and/or require user input. [31] introduces a *simulation* RDF quotient based on *triple (not node) equivalence*. (Intuitively, simulation is a unidirectional "half" of bisimulation, e.g., replace "iff" by "if" in the bisimilarity condition). For compactness, they bound the simulation; semantics is not studied, either. [3] studies simple methods for summarizing $\mathtt{D}_\mathtt{G}$, i.e. the data triples only. Data nodes are distributed in groups, e.g., a node joins a group corresponding to each type (or data property) it has in $\mathtt{G}$; this entails that some nodes belong in 0 groups, and others may belong to several, unlike the quotient summaries we consider. Implicit triple summarization is not considered.

We had demonstrated [5] and (informally) presented $\mathtt{G}_{/\mathtt{W}}$ and $\mathtt{G}_{/\mathtt{TW}}$ in a short "work in progress" paper [4], with procedural definitions (not as quotients). The shortcut was briefly described in a poster [6]. In [18], we propose a type-first summarization technique which exploits subclass hierarchies; beyond the quotient summary framework which it shares, it is orthogonal to the present work.

10 Conclusion

With the goal of obtaining concise yet informative summaries of RDF graphs, we defined property cliques, weak and strong equivalence, provided the first quotient formalization of $\mathtt{G}_{/\mathtt{W}}$, $\mathtt{G}_{/\mathtt{TW}}$, and introduced their counterparts $\mathtt{G}_{/\mathtt{S}}$ and $\mathtt{G}_{/\mathtt{TS}}$. We have established theoretically (for $\mathtt{G}_{/\mathtt{W}}$) and experimentally (for all of them) their compactness; they compress their inputs by up to $3 \cdot 10^6$.

Unlike previous equivalence relations mostly focused on *conjunctive* conditions to be satisfied by equivalent nodes, our equivalence relations support a controlled amount of *disjunction*, which makes them much more tolerant to heterogeneity. Thus, while conjunctive summaries (used as structural indexes) can help the query engine evaluate queries, our summaries are complementary; they are meant to help users and application discover the graph structure. This is the use for which they have been integrated in the LODAtlas [32] data exploration and visualization platform; their compactness was also a strong argument for chosing them.

Our second major contribution is the study of the *interplay between saturation and quotient summarization*. We introduced the *shortcut method* for summarizing a semantics-rich RDF graph without saturating it, and formally established a sufficient condition for the shortcut to hold on *any* equivalence relation, based on which we have shown that it holds for $\equiv_\mathtt{W}$, $\equiv_\mathtt{S}$ and $\equiv_\mathtt{fb}$; this translates into large performance savings when summarizing $\mathtt{G}^\infty$. Last but not least, we have presented novel, *efficient* $O(|\mathtt{G}|)$ *algorithms* for building our summaries, and for *incrementally maintaining* them in constant per-triple time under the hypothesis ($\bigstar$) (Section 7.1).

Our current work exploits our summaries for summary-based keyword search in RDF graphs; we are also experimenting with parallel summarization algorithms based on Spark.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] C. Bizer and A. Schultz. The Berlin SPARQL Benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24, 2009.

[3] S. Campinas, R. Delbru, and G. Tummarello. Efficiency and precision trade-offs in graph summary algorithms. In *IDEAS*, 2013.

[4] Š. Čebirić, F. Goasdoué, and I. Manolescu. Query-oriented summarization of RDF graphs. In *BICOD*, 2015.

[5] Š. Čebirić, F. Goasdoué, and I. Manolescu. Query-oriented summarization of RDF graphs (demonstration). *PVLDB*, 8(12), 2015.

[6] Š. Čebirić, F. Goasdoué, and I. Manolescu. A framework for efficient representative summarization of RDF graphs. In *ISWC (poster)*, 2017.

[7] C. Chen, C. X. Lin, M. Fredrikson, M. Christodorescu, X. Yan, and J. Han. Mining graph patterns efficiently via randomized summaries. *PVLDB*, 2(1), 2009.

[8] C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu. Graph OLAP: towards online analytical processing on graphs. In *ICDM*, 2008.

[9] Q. Chen, A. Lim, and K. W. Ong. $D(K)$-index: An adaptive structural summary for graph-structured data. In *SIGMOD*, 2003.

[10] M. P. Consens, V. Fionda, S. Khatchadourian, and G. Pirrò. S+EPPs: Construct and explore bisimulation summaries + optimize navigational queries; all on existing SPARQL systems (demonstration). *PVLDB*, 8(12), 2015.

[11] M. P. Consens, R. J. Miller, F. Rizzolo, and A. A. Vaisman. Exploring XML web collections with DescribeX. *TWEB*, 4(3), 2010.

[12] M. P. Consens, R. J. Miller, F. Rizzolo, and A. A. Vaisman. Exploring XML web collections with DescribeX. *ACM TWeb*, 4(3), 2010.

[13] W. Fan, J. Li, X. Wang, and Y. Wu. Query preserving graph compression. In *SIGMOD*, 2012.

[14] F. Goasdoué, I. Manolescu, and A. Roatiş. Efficient query answering against dynamic RDF databases. In *EDBT*, 2013.

[15] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, 1997.

[16] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3), 2005.

[17] S. Gurajada, S. Seufert, I. Miliaraki, and M. Theobald. Using graph summarization for join-ahead pruning in a distributed RDF engine. In *SWIM*, 2014.

[18] P. Guzewicz and I. Manolescu. Quotient RDF Summaries Based on Type Hierarchies. In *DESWeb (Data Engineering meets the Semantic Web) Workshop*, Paris, France, Apr. 2018.

[19] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *FOCS*, 1995.

[20] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth. Covering indexes for branching path queries. In *SIGMOD*, 2002.

[21] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth. Covering indexes for branching path queries. In *SIGMOD*, 2002.

[22] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *ICDE*, 2002.

[23] K. Khan, W. Nawaz, and Y. Lee. Set-based approximate approach for lossless graph summarization. *Computing*, 97(12):1185–1207, 2015.

[24] S. Khatchadourian and M. P. Consens. ExpLOD: Summary-based exploration of interlinking and RDF usage in the linked open data cloud. In *ESWC*, 2010.

[25] S. Khatchadourian and M. P. Consens. Constructing bisimulation summaries on a multi-core graph processing framework. In *GRADES*, 2015.

[26] K. LeFevre and E. Terzi. GraSS: Graph structure summarization. In *SDM*, 2010.

[27] S.-D. Lin, M.-Y. Yeh, and C.-T. Li. Sampling and summarization for social networks (tutorial), 2013.

[28] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Comput. Surv.*, 51(3):62:1–62:34, June 2018.

[29] T. Milo and D. Suciu. Index structures for path expressions. In *ICDT*, 1999.

[30] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, 2008.

[31] F. Picalausa, Y. Luo, G. H. L. Fletcher, J. Hidders, and S. Vansummeren. A structural approach to indexing triples. In *ESWC*, 2012.

[32] E. Pietriga, H. Gözükan, C. Appert, M. Destandau, Šejla Čebirić, F. Goasdoué, and I. Manolescu. Browsing linked data catalogs with LODAtlas. In *Int'l. Semantic Web Conference (ISWC), Resources track*, 2018.

[33] M. Rudolf, M. Paradies, C. Bornhövd, and W. Lehner. SynopSys: large graph analytics in the SAP HANA database through summarization. In *GRADES*, 2013.

[34] A. Schätzle, A. Neu, G. Lausen, and M. Przyjaciel-Zablocki. Large-scale bisimulation of RDF graphs. In *SWIM*, 2013.

[35] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*. ACM, 2008.

[36] T. Tran, G. Ladwig, and S. Rudolph. Managing structured and semistructured RDF data using structure indexes. *IEEE TKDE*, 25(9), 2013.

[37] W3C. Resource description framework. http://www.w3.org/RDF/.

[38] P. Zhao, J. X. Yu, and P. S. Yu. Graph indexing: Tree + delta >= graph. In *VLDB*, 2007.

For the interested reader, we include here the proofs of all the statements made in the paper. The main proofs are those of our general shortcut Theorems 1 and 2, those of `W` and `S` shortcuts (Theorems 3 and 4), and the correctness proofs for our incremental algorithms (Propositions 6 and 7). The other claims proved here are used in these main proofs.

A Proof of Proposition 1

Proof. First, note that any two weak summary nodes n_1, n_2 cannot be targets of the same data property. Indeed, if such a data property p existed, let TC be the target clique it belongs to. By the definition of the weak summary, n_1 corresponds to a set of (disjoint) target cliques STC_1, which includes TC, and a set of disjoint source cliques SSC_1. Similarly, n_2 corresponds to a set of (disjoint) target cliques STC_2, which includes TC, and a set of disjoint source cliques SSC_2. The presence of TC in STC_1 and STC_2 contradicts the fact that different equivalence classes of G nodes correspond to disjoint sets of target cliques. The same holds for the sets of properties of which weak summary nodes are sources. Thus, any data property has at most one source and at most one target in $\mathtt{G}_{/\mathtt{W}}$. Further, by the definition of the summary as a quotient, every data property present in G also appears in the summary. Thus, there is exactly one p-labeled edge in $\mathtt{G}_{/\mathtt{W}}$ for every data property in G. □

B Proof of Proposition 2

Proof. If two $\mathtt{G}_\mathtt{S}$ distinct nodes had the same source and the same target clique, they would be strongly equivalent. This cannot be the case in a quotient summary obtained through $\equiv_\mathtt{S}$, since by definition, such a summary has one node for each $\equiv_\mathtt{S}$ equivalence class. Thus, any two distinct $\mathtt{G}_\mathtt{S}$ have distinct source cliques and/or distinct target cliques.

Now, let m be a $\mathtt{G}_\mathtt{S}$ node, and $S_m = f_\mathtt{S}^{-1}(m)$ be the set of all G nodes represented by m. By the definition of a quotient summary, m must be the target (resp. the source) of an edge carrying each of the labels on the edges entering (resp. going out of) any node $n \in S_m$. Thus, m is source of all the properties in the source clique shared by the nodes in S_m, and is target of all the properties in the target clique shared by the nodes in S_m. Thus, m has the source and target clique of any node from S_m; this concludes our proof. □

C Proof of Proposition 3

Proof. By definition of an equivalence relation between nodes, every class (resp. property) node is only equivalent to itself, and is represented by itself (i.e., has same label in a graph and in the corresponding summary). Therefore, by definition of a quotient graph using such a node equivalence relation, every edge between class and/or property nodes begets an edge with same label in the graph summary. As schema edges only connects such nodes, schema triples are copied by summarization from a graph to its summary. □

D Proof of Theorem 1

Proof. We first show that an homomorphism can be established from the node sets of $\mathtt{G}^\infty$ to that of $(\mathtt{G}_{/\equiv})^\infty$.

Observe that RDF saturation with RDFS constraints only adds edges between graph nodes, but does not add nodes. Thus, a node n is in $\mathtt{G}^\infty$ iff n is in G. Further, by the definition of our quotient-based summaries, n is in G iff $f_\equiv(n)$ is in $\mathtt{G}_{/\equiv}$. Finally, again by the definition of saturation, $f_\equiv(n)$ is in $\mathtt{G}_{/\equiv}$ iff $f_\equiv(n)$ is in $(\mathtt{G}_{/\equiv})^\infty$.

Therefore, every $\mathtt{G}^\infty$ node n maps the $f_\equiv(n)$ $(\mathtt{G}_{/\equiv})^\infty$ node (*).

Next, we show that there is a one-to-one mapping between $\mathtt{G}^\infty$ edges and those of $(\mathtt{G}_{/\equiv})^\infty$.

If $n_1\ p\ n_2$ is an edge in $\mathtt{G}^\infty$, at least one of the following two situations holds:

- $n_1\ p\ n_2$ is an edge in $\mathtt{G}$. This holds iff $f_{\equiv}(n_1)\ p\ f_{\equiv}(n_2)$ is an edge in $\mathtt{G}_{/\equiv}$, by definition of an RDF summary. Finally, if $f_{\equiv}(n_1)\ p\ f_{\equiv}(n_2)$ is an edge in $\mathtt{G}_{/\equiv}$, then $f_{\equiv}(n_1)\ p\ f_{\equiv}(n_2)$ is also an edge in $(\mathtt{G}_{/\equiv})^{\infty}$.
- $n_1\ p'\ n_2$ is an edge in $\mathtt{G}$, and p' *subproperty* p is in $\mathtt{S}_{\mathtt{G}}^{\infty}$, thus $n_1\ p\ n_2$ is produced by saturation in $\mathtt{G}^{\infty}$. In this case, we show similarly to the preceding item that $f_{\equiv}(n_1)\ p'\ f_{\equiv}(n_2)$ is an edge in $(\mathtt{G}_{/\equiv})^{\infty}$, hence $f_{\equiv}(n_1)\ p\ f_{\equiv}(n_2)$ is also an edge added to $(\mathtt{G}_{/\equiv})^{\infty}$ by saturation, since $(\mathtt{G}_{/\equiv})^{\infty}$ and $\mathtt{G}^{\infty}$ have the same (saturated) schema triples (Proposition 3).

If n_1 *type* c is an edge in $\mathtt{G}^{\infty}$, at least one of the following two situations holds:

- n_1 *type* c is an edge in $\mathtt{G}$. This holds iff $f_{\equiv}(n_1)$ *type* c is an edge in $\mathtt{G}_{/\equiv}$, by definition of an RDF summary (recall that $f_{\equiv}(c) = c$ for classes). Finally, if $f_{\equiv}(n_1)$ *type* c is an edge in $\mathtt{G}_{/\equiv}$, then $f_{\equiv}(n_1)$ *type* c is also an edge in $(\mathtt{G}_{/\equiv})^{\infty}$.
- $n_1\ p\ n_2$ is an edge in $\mathtt{G}$ and p *domain* c (or p *range* c) is in $\mathtt{S}_{\mathtt{G}}^{\infty}$, thus n_1 *type* c is produced by saturation in $\mathtt{G}^{\infty}$. In this case, we show similarly as above that $f_{\equiv}(n_1)\ p\ f_{\equiv}(n_2)$ is an edge in $(\mathtt{G}_{/\equiv})^{\infty}$, hence $f_{\equiv}(n_1)$ *type* c is also an edge added to $(\mathtt{G}_{/\equiv})^{\infty}$ by saturation, since $(\mathtt{G}_{/\equiv})^{\infty}$ and $\mathtt{G}^{\infty}$ have the same (saturated) schema triples (Proposition 3).

Therefore, every $\mathtt{G}^{\infty}$ edge $n_1\ p\ n_2$ (resp. n_1 *type* c) maps into the $(\mathtt{G}_{/\equiv})^{\infty}$ edge $f_{\equiv}(n_1)\ p\ f_{\equiv}(n_2)$ (resp. $f_{\equiv}(n_1)$ *type* c) (**).

From (*) and (**), it follows that f is an homomorphism from $\mathtt{G}^{\infty}$ to $(\mathtt{G}_{/\equiv})^{\infty}$. □

E Proof of Theorem 2

Proof. We start by introducing some **notations** (see Figure 10). Let f_1 be the representation function from $\mathtt{G}^{\infty}$ into $(\mathtt{G}^{\infty})_{/\equiv}$, and f_2 be the representation function from $(\mathtt{G}_{/\equiv})^{\infty}$ into $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$.

Let the function φ be a function from the $(\mathtt{G}^{\infty})_{/\equiv}$ nodes to the $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ nodes defined as: $\varphi(f_1(n)) = f_2(f(n))$ for n any $\mathtt{G}^{\infty}$ node.

Suppose that for every pair (n_1, n_2) of $\mathtt{G}$ nodes, $n_1 \equiv n_2$ in $\mathtt{G}^{\infty}$ iff $f(n_1) \equiv f(n_2)$ in $(\mathtt{G}_{/\equiv})^{\infty}$ holds. Let us show that this condition suffices to ensure $(\mathtt{G}^{\infty})_{/\equiv} \equiv ((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ holds, i.e., the φ function defines an isomorphism from $(\mathtt{G}^{\infty})_{/\equiv}$ to $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$.

First, let us show that φ is a bijection from all the $(\mathtt{G}^{\infty})_{/\equiv}$ nodes to all the $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ nodes. Since for every pair n_1, n_2 of $\mathtt{G}^{\infty}$ nodes, $n_1 \equiv n_2$ iff $f(n_1) \equiv f(n_2)$ in $(\mathtt{G}_{/\equiv})^{\infty}$, it follows that $(\mathtt{G}^{\infty})_{/\equiv}$ and $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ have the same number of nodes (*).

Further, a given node n in $(\mathtt{G}^{\infty})_{/\equiv}$ represents a set of equivalent nodes $n_1, \ldots, n_k$ from $\mathtt{G}^{\infty}$. By hypothesis, $n_1 \equiv \cdots \equiv n_k$ in $\mathtt{G}^{\infty}$ iff $f(n_1) \equiv \cdots \equiv f(n_k)$ in $\mathtt{G}^{\infty}_{/\equiv}$ holds. Hence, every node $n = f_1(n_1) = \cdots = f_1(n_k)$ of $(\mathtt{G}^{\infty})_{/\equiv}$ maps to a distinct node $n' = f_2(f(n_1)) = \cdots = f_2(f(n_k))$ in $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ (**).

Similarly, a given node n' in $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ represents a set of equivalent nodes $n'_1 = f(n_1), \ldots, n'_k = f(n_k)$ in $(\mathtt{G}_{/\equiv})^{\infty}$. By hypothesis, $f(n_1) \equiv \cdots \equiv f(n_k)$ in $\mathtt{G}^{\infty}_{/\equiv}$ iff $n_1 \equiv \cdots \equiv n_k$ in $\mathtt{G}^{\infty}$ holds. Hence, every node $n' = f_2(f(n_1)) = \cdots = f_2(f(n_k))$ in $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ maps to a distinct node $n = f_1(n_1) = \cdots = f_1(n_k)$ of $(\mathtt{G}^{\infty})_{/\equiv}$ (***).

From (*), (**) and (***), it follows that φ is a bijective function from all the $(\mathtt{G}^{\infty})_{/\equiv}$ nodes to all the $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$ nodes.

Now, let us show that φ defines an isomorphism from $(\mathtt{G}^{\infty})_{/\equiv}$ to $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$.

For every edge $n'_1\ p\ n'_2$ in $(\mathtt{G}^{\infty})_{/\equiv}$, by definition of an RDF summary, there exists an edge $n_1\ p\ n_2$ in $\mathtt{G}^{\infty}$ such that $n'_1\ p\ n'_2 = f_1(n_1)\ p\ f_1(n_2)$. Figure 16 illustrates the discussion. Further, if $n_1\ p\ n_2$ is in $\mathtt{G}^{\infty}$, then $f(n_1)\ p\ f(n_2)$ is in $(\mathtt{G}_{/\equiv})^{\infty}$ (Theorem 1), hence $f_2(f(n_1))\ p\ f_2(f(n_2))$ is in $((\mathtt{G}_{/\equiv})^{\infty})_{/\equiv}$. Therefore,

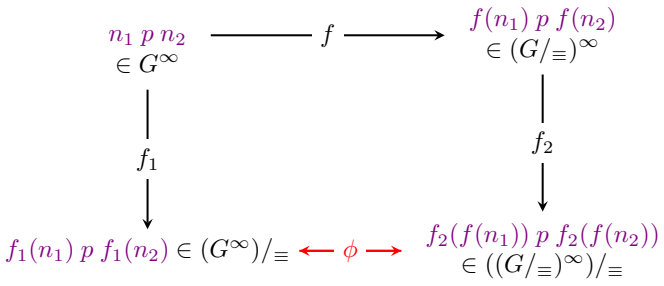


Figure 16: Diagram for the proof of Theorem 2.

- since for every $f_1(n_1)\ p\ f_1(n_2)$ edge in $(\mathtt{G}^\infty)_{/\equiv}$, there is an edge $f_2(f(n_1))\ p\ f_2(f(n_2))$ in $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$, and
- since $\varphi(f_1(n)) = f_2(f(n))$, for n any $\mathtt{G}^\infty$ node, is a bijective function from all $(\mathtt{G}^\infty)_{/\equiv}$ nodes to all $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$ nodes,
- it follows that $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$ contains the image of all $(\mathtt{G}^\infty)_{/\equiv}\ f_1(n_1)\ p\ f_1(n_2)$ triples through φ (*).

Now, for every edge $n_1''\ p\ n_2''$ in $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$, by definition of an RDF summary, there exists an edge $n_1'\ p\ n_2'$ in $(\mathtt{G}_{/\equiv})^\infty$ such that $n_1''\ p\ n_2'' = f_2(n_1')\ p\ f_2(n_2')$. Hence, by Theorem 1, there exists an edge $n_1\ p\ n_2$ in $\mathtt{G}^\infty$ such that $n_1'\ p\ n_2' = f(n_1)\ p\ f(n_2)$. Moreover, since $n_1\ p\ n_2$ is in $\mathtt{G}^\infty$, $f_1(n_1)\ p\ f_1(n_2)$ is in $(\mathtt{G}^\infty)_{/\equiv}$. Therefore, since for every $f_2(f(n_1))\ p\ f_2(f(n_2))$ edge in $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$, there is an edge $f_1(n_1)\ p\ f_1(n_2)$ in $(\mathtt{G}^\infty)_{/\equiv}$, and since $\varphi(f_1(n)) = f_2(f(n))$, for n any $\mathtt{G}^\infty$ node, is a bijective function from all $(\mathtt{G}^\infty)_{/\equiv}$ nodes to all $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$ nodes, $(\mathtt{G}^\infty)_{/\equiv}$ contains the image of all $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}\ n_1''\ p\ n_2''$ triples through φ^{-1} (**).

Similarly, for every edge $n_1'\ \tau\ c$ in $(\mathtt{G}^\infty)_{/\equiv}$, by definition of an RDF summary, there exists an edge $n_1\ \tau\ c$ in $\mathtt{G}^\infty$ such that $n_1'\ \tau\ c = f_1(n_1)\ \tau\ c$. Further, if $n_1\ \tau\ c$ is in $\mathtt{G}^\infty$, then $f(n_1)\ \tau\ c$ is in $(\mathtt{G}_{/\equiv})^\infty$ (Theorem 1), hence $f_2(f(n_1))\ \tau\ c$ is in $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$. Therefore,

- since for every $f_1(n_1)\ \tau\ c$ edge in $(\mathtt{G}^\infty)_{/\equiv}$, there is an edge $f_2(f(n_1))\ \tau\ c$ in $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$, and
- since $\varphi(f_1(n)) = f_2(f(n))$, for n any $\mathtt{G}^\infty$ node, is a bijective function from all $(\mathtt{G}^\infty)_{/\equiv}$ nodes to all $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$ nodes,
- it follows that $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$ contains the image of all $(\mathtt{G}^\infty)_{/\equiv}\ f_1(n_1)\ \tau\ c$ triples through φ (*').

Now, for every edge $n_1''\ \tau\ c$ in $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$, by definition of an RDF summary, there exists an edge $n_1'\ \tau\ c$ in $(\mathtt{G}_{/\equiv})^\infty$ such that $n_1''\ \tau\ c = f_2(n_1')\ \tau\ c$. Hence, by Theorem 1, there exists an edge $n_1\ \tau\ c$ in $\mathtt{G}^\infty$ such that $n_1'\ \tau\ c = f(n_1)\ \tau\ c$. Moreover, since $n_1\ \tau\ c$ is in $\mathtt{G}^\infty$, $f_1(n_1)\ \tau\ c$ is in $(\mathtt{G}^\infty)_{/\equiv}$. Therefore, since for every $f_2(f(n_1))\ \tau\ c$ edge in $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$, there is an edge $f_1(n_1)\ \tau\ c$ in $(\mathtt{G}^\infty)_{/\equiv}$, and since $\varphi(f_1(n)) = f_2(f(n))$, for n any $\mathtt{G}^\infty$ node, is a bijective function from all $(\mathtt{G}^\infty)_{/\equiv}$ nodes to all $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$ nodes, $(\mathtt{G}^\infty)_{/\equiv}$ contains the image of all $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}\ n_1''\ \tau\ c$ triples through φ^{-1} (**').

From (*) and (**), and, (*') and (**'), it follows that φ defines an isomorphism from $(\mathtt{G}^\infty)_{/\equiv}$ to $((\mathtt{G}_{/\equiv})^\infty)_{/\equiv}$. □

F Proof of Lemma 1

Proof. We prove the lemma only for source cliques; the proof for the target cliques is very similar.

1. Any resource $r \in \mathtt{G}$ having two data properties also has them in $\mathtt{G}^\infty$; thus, any data properties in the same source clique in $\mathtt{G}$ are also in the same source clique in $\mathtt{G}^\infty$. The unicity of C^∞ is ensured by the fact that the source cliques of $\mathtt{G}^\infty$ are by definition disjoint.

2. C_1^+ and C_2^+ intersect on property $\mathtt{p}$ iff there exist some $\mathtt{p}_1 \in C_1$ and $\mathtt{p}_2 \in C_2$ which are specializations of the same $\mathtt{p}$ (one, but not both, may also be $\mathtt{p}$ itself). Independently, we know that there exist $r_1, r_2 \in \mathtt{G}$ such that r_1 has $\mathtt{p}_1$ and r_2 has $\mathtt{p}_2$; in $\mathtt{G}^\infty$, r_1 has $\mathtt{p}_1$ and $\mathtt{p}$, thus these two properties are in the same $\mathtt{G}^\infty$ clique. Similarly, r_2 has $\mathtt{p}_2$ and $\mathtt{p}$, which ensures that $\mathtt{p}$ is also in the same $\mathtt{G}^\infty$ source clique.

3. Let $\{\mathtt{p}_1, \ldots, \mathtt{p}_k\}$ be the data properties that appear both in $\mathtt{G}$ and in C^∞; it follows from the saturation rules and the definition of cliques, that $k > 0$. For $1 \leq i \leq k$, let C_i be the $\mathtt{G}$ source clique comprising $\mathtt{p}_i$. Applying lemma point 1., $C_i \subseteq C^\infty$ for each $1 \leq i \leq k$. Further, it is easy to see that $C_i^+ \subseteq C^\infty$, since any property that saturation adds to C_i^+ is also added by saturation to C^∞. Thus, $\bigcup_{1 \leq i \leq k} C_i^+ \subseteq C^\infty$.

   Let us now show that $C^\infty \subseteq \bigcup_{1 \leq i \leq k} C_i^+$. Let $\mathtt{p} \in C^\infty$ be a data property, then there exists a resource r having $\mathtt{p}$ in $\mathtt{G}^\infty$. Then, in $\mathtt{G}$, r has a property $\mathtt{p}'$ which is either $\mathtt{p}$, or is such that $\mathtt{p}'$ *subproperty* $\mathtt{p}$ in $\mathtt{G}^\infty$. Then, in $\mathtt{G}^\infty$, r has both $\mathtt{p}$ and $\mathtt{p}'$, which entails that $\mathtt{p}' \in C^\infty$. Therefore, $\mathtt{p}'$ is a data property occurring both in C^∞ and in $\mathtt{G}$, therefore $\mathtt{p}'$ is one of the properties $\mathtt{p}_i$, for some $1 \leq i \leq k$, that is, $\mathtt{p}' \in C_i$, and accordingly, $\mathtt{p} \in C_i^+$ due to $\mathtt{p}'$ *subproperty* $\mathtt{p}$.

   Thus, any data property $\mathtt{p} \in C^\infty$ is part of some C_i^+.

   We must still show that the saturated cliques intersect. If $k = 1$ the statement is trivially true. Suppose $k \geq 2$ and the statement is false. Let $\mathcal{C}$ denote the set $\{C_1, \ldots, C_m\}$; the cliques in $\mathcal{C}$ are pairwise disjoint by definition. Let $\mathcal{I} \subseteq \mathcal{C}$ be a *maximal* subset of $\mathcal{C}$ cliques such that the saturations of $\mathcal{I}$ cliques all intersect (directly or indirectly). Let $\mathcal{J} = \mathcal{C} \setminus \mathcal{I}$ be the complement of $\mathcal{I}$; if the last part of 4. is false, $\mathcal{J}$ is not empty. We denote $\mathcal{I}^+$, respectively $\mathcal{J}^+$, the set of the saturated cliques from $\mathcal{I}$, resp. $\mathcal{J}$.

   No data property $\mathtt{p}_i$ from $\mathcal{I}^+$ can be source-related in $\mathtt{G}^\infty$ to any data property $\mathtt{p}_j$ from $\mathcal{J}^+$. This is because source-relatedness requires a resource r having in $\mathtt{G}^\infty$ both $\mathtt{p}_i$ and a property $\mathtt{p}$ source-related to $\mathtt{p}_j$. If such a property $\mathtt{p}$ existed, it would belong both to $\mathcal{I}^+$ (since $\mathtt{p}$ has a common source with $\mathtt{p}_i$) and to $\mathcal{J}^+$ (since $\mathtt{p}$ is source-related to $\mathtt{p}_j$); or, $\mathcal{I}^+$ and $\mathcal{J}^+$ have no property in common.

   The lack of source-relatedness in $\mathtt{G}^\infty$ between $\mathtt{p}_i$ and $\mathtt{p}_j$ chosen as above contradicts the hypothesis that they are part of the same source clique of $\mathtt{G}^\infty$, namely C^∞.

4. The statement follows quite directly as a consequence of the previous one, concluding our proof.

□

G Proof of Lemma 2

Proof. We prove the lemma for target-related properties.

"Only if": If data properties are target-related in $(\mathtt{G}_{/\mathtt{W}})^\infty$, then they belong to a same target clique $TC_{\mathtt{W}}^\infty$ in $(\mathtt{G}_{/\mathtt{W}})^\infty$.

By Lemma 1, point 3, it follows that $TC_{\mathtt{W}}^\infty$ is the union of the saturations of a set of $\mathtt{G}_{/\mathtt{W}}$ cliques $(TC_{\mathtt{W}}^1)^+, (TC_{\mathtt{W}}^2)^+, \ldots, (TC_{\mathtt{W}}^m)^+$. Then:

- For every $1 \leq j \leq m$:
  - $TC^j_{\mathtt{W}}$ is the target clique of a $\mathtt{G}_{/\mathtt{W}}$ node n^j;
  - n^j represents a set of weakly-equivalent $\mathtt{G}$ resources, which are targets only of properties in $TC^j_{\mathtt{W}}$. Thus, the properties in $TC^j_{\mathtt{W}}$ are target-related in $\mathtt{G}$.
  - Thus, in $\mathtt{G}^\infty$, also, the properties in $TC^j_{\mathtt{W}}$ are target-related.
  - From this and the definition of a saturated graph and of a saturated target clique, it follows that the properties from $(TC^j_{\mathtt{W}})^+$ are target-related in $\mathtt{G}^\infty$.
- Further, still by Lemma 1, point 4, each $(TC^j_{\mathtt{W}})^+$ intersects at least another $(TC^l_{\mathtt{W}})^+$ for $1 \leq l \neq j < m$, thus the target properties in all the $(TC^j_{\mathtt{W}})^+$ for $1 \leq j \leq m$, and in particular $\mathtt{p}$, are target-related to each other in $\mathtt{G}^\infty$. Thus, $\mathtt{p}$ is target-related in $\mathtt{G}^\infty$ to all properties from $TC^\infty_{\mathtt{W}}$.

"If": if data properties are target-related in $\mathtt{G}^\infty$, then they belong to a same target clique TC^∞ in $\mathtt{G}^\infty$. Let $n_1, \ldots, n_k$ be the set of all $\mathtt{G}$ resources which are values of some properties in TC^∞. By definition of an RDF summary and Theorem 1, each summary representative $f(n_i)$ of n_i, for $1 \leq i \leq k$, is at least the object of the same properties as n_i, hence all the properties of TC^∞ in $\mathtt{G}^\infty$ are target-related in $(\mathtt{G}_{/\mathtt{W}})^\infty$. □

H Proof of Proposition 4

Proof. Recall from Lemma 1 that:

$$SC^\infty_{\mathtt{W}} = (SC^1_{\mathtt{W}})^+ \cup (SC^2_{\mathtt{W}})^+ \cup \ldots \cup (SC^m_{\mathtt{W}})^+ \text{ and}$$
$$TC^\infty_{\mathtt{W}} = (TC^1_{\mathtt{W}})^+ \cup (TC^2_{\mathtt{W}})^+ \cup \ldots \cup (TC^n_{\mathtt{W}})^+$$

for some $\mathtt{G}_{/\mathtt{W}}$ source cliques $SC^1_{\mathtt{W}}, \ldots, SC^m_{\mathtt{W}}$ and target cliques $TC^1_{\mathtt{W}}, \ldots, TC^m_{\mathtt{W}}$.

Lemma 2 ensures that the data properties in $(SC^1_{\mathtt{W}})^+ \cup \ldots \cup (SC^m_{\mathtt{W}})^+$ are related in $\mathtt{G}^\infty$, and those of $(TC^1_{\mathtt{W}})^+ \cup \ldots \cup (TC^n_{\mathtt{W}})^+$ are related in $\mathtt{G}^\infty$.

Moreover, $n_{\mathtt{W}}$ was created in $\mathtt{G}_{/\mathtt{W}}$ from a set of weakly-equivalent $\mathtt{G}$ nodes having as source clique one among $SC^1_{\mathtt{W}}, \ldots, SC^m_{\mathtt{W}}$ and as target clique one among $TC^1_{\mathtt{W}}, \ldots, TC^n_{\mathtt{W}}$. In $\mathtt{G}^\infty$, these nodes connect the data properties of $SC^\infty_{\mathtt{W}}$ with those of $TC^\infty_{\mathtt{W}}$. □

I Proof of Theorem 3

Proof. We show that $\mathtt{W}$ summaries enjoy the sufficient condition for completenes stated in Theorem 2, i.e., given two nodes n_1, n_2 in $\mathtt{G}^\infty$, f the representation homomorphism corresponding to the weak-equivalence relation $\equiv_{\mathtt{W}}$, and $f(n_1)$, $f(n_2)$ the images of n_1, n_2 in $(\mathtt{G}_{/\mathtt{W}})^\infty$ through f (recall Theorem 1), it holds that: $n_1 \equiv_{\mathtt{W}} n_2$ in $\mathtt{G}^\infty$ iff $f(n_1) \equiv_{\mathtt{W}} f(n_2)$ in $(\mathtt{G}_{/\equiv_{\mathtt{W}}})^\infty$.

"Only if": $n_1 \equiv_{\mathtt{W}} n_2$ in $\mathtt{G}^\infty$ iff they are connected by an alternating chain of source and target cliques of $\mathtt{G}^\infty$ (as shown in Figure 4); to reuse that figure for the current proof, let us use n_{2k} to denote the n_2 of the current lemma statement. Note that $\mathtt{G}^\infty$ only adds triples not nodes, thus all the nodes shown in the figure also exist in $\mathtt{G}$. Now, let us consider the $\mathtt{G}_{\mathtt{W}}$ nodes $f(n_1), f(n_2), \ldots, f(n_{2k})$ obtained by applying the representation function f on $n_1, \ldots, n_{2k}$. By Theorem 1, f is also a homomorphism from $\mathtt{G}^\infty$ to $(\mathtt{G}_{\mathtt{W}})^\infty$, therefore any incoming (outgoing) edge into (from) a node n_j of $\mathtt{G}^\infty$ is also incoming (resp. outgoing) into (from) the respective node $f(n_j)$ of $(\mathtt{G}_{\mathtt{W}})^\infty$. As a consequence, we can reproduce

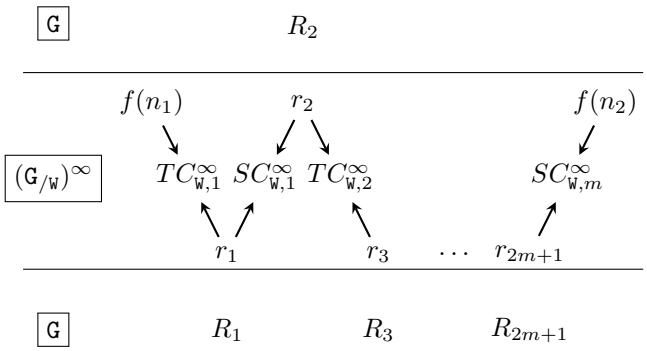


Figure 17: Sketch for sufficient condition for weak summary completeness.

the alternating clique structure into $(\mathtt{G}_{\mathtt{W}})^\infty$, which suffices to make $f(n_1)$ and $f(n_{2k})$ weakly equivalent in $(\mathtt{G}_{\mathtt{W}})^\infty$.

"If": $f(n_1) \equiv_{\mathtt{W}} f(n_2)$ in $(\mathtt{G}_{/\mathtt{W}})^\infty$ iff they are connected by an alternating chain of source and target cliques in $(\mathtt{G}_{/\equiv_{\mathtt{W}}})^\infty$. Assume w.l.o.g. that the chain is as shown in Figure 17, that is, of the form:

- $f(n_1)$ shares a target clique $TC^\infty_{\mathtt{W},1}$ with r_1
- $r_1(TC^\infty_{\mathtt{W},1}, SC^\infty_{\mathtt{W},1})$
- $r_2(TC^\infty_{\mathtt{W},2}, SC^\infty_{\mathtt{W},1}), \ldots,$
- $r_{2m+1}(TC^\infty_{\mathtt{W},m-1}, SC^\infty_{\mathtt{W},m})$, and $f(n_2)$ has the source clique $SC^\infty_{\mathtt{W},m}$

The alternating chain starts with a target clique and ends with a source clique (of course three other combinations are possible). In the chain, each resource is either:

- $r_{2i+1}(TC^\infty_{\mathtt{W},i+1}, SC^\infty_{\mathtt{W},i+1})$ or
- $r_{2i+2}(TC^\infty_{\mathtt{W},i+2}, SC^\infty_{\mathtt{W},i+1})$

for some $0 \leq i < m$. Every r_{2i+1} and r_{2i+2} resource is a node from $(\mathtt{G}_{/_{\mathtt{W}}})^\infty$, thus a node from $\mathtt{G}_{/\mathtt{W}}$ (because saturating $\mathtt{G}_{/\mathtt{W}}$ does not create nodes). For a given r_j, let R_j be the set of weakly-equivalent $\mathtt{G}$ resources from which r_j was created; all resources in R_j are by definition weakly equivalent in $\mathtt{G}$, and this also holds in $\mathtt{G}^\infty$.

By Proposition 4, $TC^\infty_{\mathtt{W},1}$ is also a target clique of $\mathtt{G}^\infty$, and it must be the target clique of n_1 in $\mathtt{G}^\infty$ (because of the f homomorphism from $\mathtt{G}^\infty$ into $(\mathtt{G}_{/\mathtt{W}})^\infty$ ensured by Theorem 1). Similarly, $SC^\infty_{\mathtt{W},m}$ must be a source clique in $\mathtt{G}^\infty$ and in particular the source clique of n_2.

In $\mathtt{G}^\infty$, n_1 shares its target clique $TC^\infty_{\mathtt{W},1}$ with the nodes in R_1, thus n_1 is weakly-equivalent to any node from R_1.

Further, by Proposition 4, if the node r_1 has the target clique $TC^\infty_{\mathtt{W},1}$ and the source clique $SC^\infty_{\mathtt{W},1}$ in $(\mathtt{G}_{/\mathtt{W}})^\infty$, then the $\mathtt{G}^\infty$ node whose target clique is $TC^\infty_{\mathtt{W},1}$ must also have the source clique $SC^\infty_{\mathtt{W},1}$ in $\mathtt{G}^\infty$. (Proposition 4 also ensures that a node in $\mathtt{G}^\infty$ has $SC^\infty_{\mathtt{W},1}$ as its source clique.)

If the alternating chain is long enough to comprise r_2 (that is: if the chain does not degenerate in a single node), that corresponds to the set R_2 of $\mathtt{G}$ nodes which, in $\mathtt{G}^\infty$, have the source clique $SC^\infty_{\mathtt{W},1}$, therefore they are weakly equivalent to all nodes from R_1 which have the same source clique. Thus, n_1 is weakly equivalent in $\mathtt{G}^\infty$ to the nodes from R_1 and R_2.

The above reasoning can be applied on each edge in the alternating chain, extending weak equivalence from n_1 through all the R_j sets until n_2. □

J Proof of Lemma 3

Proof. "If": if data properties are target-related in $\mathtt{G}^\infty$, then they belong to a same target clique TC^∞ in $\mathtt{G}^\infty$. Let $n_1, \ldots, n_k$ be the set of all $\mathtt{G}$ resources which are values of some properties in TC^∞. By definition of an RDF summary and Theorem 1, each image $f_\mathtt{S}(n_i)$ of n_i, for $1 \leq i \leq k$ is at least the object of the same properties as n_i, hence all the properties of TC^∞ in $\mathtt{G}^\infty$ are target-related in $(\mathtt{G}_{/\mathtt{S}})^\infty$.

"Only If": if two data properties p_1 and p_2 are target-related in $(\mathtt{G}_{/\mathtt{S}})^\infty$, then they belong to a same target clique $TC^{\mathtt{S},\infty}$, in which they are at distance $n \geq 0$, i.e., they are target-related because of a set $\bigcup_{i=0}^{n}\{r_{i+1}\}$ of nodes which all have the target clique $TC^{\mathtt{S},\infty}$. In $\mathtt{G}_{/\mathtt{S}}$, each such r_{i+1} has a target clique $TC_i^\mathtt{S} \subseteq TC^{\mathtt{S},\infty}$, moreover each r_{i+1} results from a set of $\mathtt{G}$ nodes $n_{i+1}^j, j \geq 1$, which by definition of a strong RDF summary, have all the source clique $TC_i^\mathtt{S}$. Hence, every such n_{i+1}^j node has target clique $TC^{\mathtt{S},\infty}$ in $\mathtt{G}^\infty$ (since $\mathtt{G}$ and $\mathtt{G}_{/\mathtt{S}}$ have the same schema), in which p_1 and p_2 are target related. □

K Proof of Proposition 5

Proof. Recall from Lemma 1 that:

$$SC_\mathtt{S}^\infty = (SC_\mathtt{S}^1)^+ \cup (SC_\mathtt{S}^2)^+ \cup \ldots \cup (SC_\mathtt{S}^m)^+ \text{ and}$$
$$TC_\mathtt{S}^\infty = (TC_\mathtt{S}^1)^+ \cup (TC_\mathtt{S}^2)^+ \cup \ldots \cup (TC_\mathtt{S}^n)^+$$

for some $\mathtt{S}_\mathtt{G}$ source cliques $SC_\mathtt{S}^1, \ldots, SC_\mathtt{S}^m$ and target cliques $TC_\mathtt{S}^1, \ldots, TC_\mathtt{S}^n$.

Lemma 3 ensures that the data properties in $(SC_\mathtt{S}^1)^+ \cup \ldots \cup (SC_\mathtt{S}^m)^+$ are related in $\mathtt{G}^\infty$, and those of $(TC_\mathtt{S}^1)^+ \cup \ldots \cup (TC_\mathtt{S}^n)^+$ are related in $\mathtt{G}^\infty$.

Moreover, $n_\mathtt{S}$ was created in $\mathtt{S}_\mathtt{G}$ from a set of strongly-equivalent $\mathtt{G}$ nodes all sharing a source clique $SC_\mathtt{S}^i$, for $1 \leq i \leq m$, and all sharing a target clique $TC_\mathtt{S}^j$, for some $1 \leq j \leq n$. Thus, in $\mathtt{G}^\infty$, these nodes connect the data properties of $SC_\mathtt{S}^\infty$ with those of $TC_\mathtt{S}^\infty$. □

L Proof of Theorem 4

Proof. "Only if" follows directly from Theorem 1.

To prove "if", note that $f(n_1) \equiv_\mathtt{S} f(n_2)$ in $(\mathtt{S}_\mathtt{G})^\infty$ iff they have the same source clique $SC_\mathtt{S}^\infty$ and the same target clique $TC_\mathtt{S}^\infty$ in $(\mathtt{S}_\mathtt{G})^\infty$. By Proposition 5, $TC_\mathtt{S}^\infty$ is also a target clique of $\mathtt{G}^\infty$, and it must be the target clique of n_1 and n_2 in $\mathtt{G}^\infty$ (because of the f homomorphism from $\mathtt{G}^\infty$ into $(\mathtt{S}_\mathtt{G})^\infty$ ensured by Theorem 1). Similarly, $SC_\mathtt{S}^\infty$ is a source clique in $\mathtt{G}^\infty$, and in particular the source clique of n_1 and n_2.

Thus, $n_1 \equiv_\mathtt{S} n_2$ in $\mathtt{G}^\infty$. □

M Proof of Theorem 7

Proof. We first prove the claim for $\equiv_\mathtt{fw}$.

We show this result using the sufficient condition stated in Theorem 2. That is, $n_1 \equiv_\mathtt{fw} n_2$ in $\mathtt{G}^\infty$ holds iff $f(n_1) \equiv_\mathtt{fw} f(n_2)$ in $(\mathtt{G}_{/\equiv_\mathtt{fw}})^\infty$ holds.

This holds for class nodes and for property nodes since, by definition, they are only equivalent to themselves through some RDF node equivalence relation.

Now, consider two data nodes n_1, n_2 in $\mathtt{G}^\infty$ such that $n_1 \equiv_\mathtt{fw} n_2$ in $\mathtt{G}^\infty$, and let us show that $f(n_1) \equiv_\mathtt{fw} f(n_2)$ in $(\mathtt{G}_{/\equiv_\mathtt{fw}})^\infty$.

If $n_1 \equiv_\mathtt{fw} n_2$ holds in $\mathtt{G}^\infty$, then for every triple n_1 p m_1 there exists a triple n_2 p m_2 such that $m_1 \equiv_\mathtt{fw} m_2$ holds, and conversely for every triple n_2 p m_2 there exists a triple n_1 p m_1 such that $m_1 \equiv_\mathtt{fw} m_2$ holds.

Let $\mathcal{P}^{\infty}_{n_1,n_2\to m_1,m_2}$ be the set of outgoing properties from n_1 to m_1 and from n_2 to m_2 in $\mathtt{G}^{\infty}$.

In $\mathtt{G}$, the set of outgoing properties from n_1 to m_1, denoted $\mathcal{P}_{n_1\to m_1}$ is a subset of $\mathcal{P}^{\infty}_{n_1,n_2\to m_1,m_2}$, since by definition the saturation of a graph only adds edges; similarly, in $\mathtt{G}$, the set of outgoing properties from n_2 to m_2, denoted $\mathcal{P}_{n_2\to m_2}$ is a subset of $\mathcal{P}^{\infty}_{n_1,n_2\to m_1,m_2}$, which may be different from $\mathcal{P}_{n_1\to m_1}$.

By definition of a $\equiv_{\mathtt{fw}}$-summary, the set of outgoing properties from $f(n_1)$ to $f(m_1)$ in $\mathtt{G}_{/\equiv_{\mathtt{fw}}}$ is exactly $\mathcal{P}_{n_1\to m_1}$ and similarly the set of outgoing properties from $f(n_2)$ to $f(m_2)$ in $\mathtt{G}_{/\equiv_{\mathtt{fw}}}$ is exactly $\mathcal{P}_{n_2\to m_2}$.

Since $\mathtt{G}$ and $\mathtt{G}_{/\equiv_{\mathtt{fw}}}$ have the same schema (Property 3), it follows that in $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$, the set of outgoing properties from $f(n_1)$ to $f(m_1)$, and from $f(n_2)$ to $f(m_2)$, is exactly $\mathcal{P}^{\infty}_{n_1,n_2\to m_1,m_2}$ (data edges can only be added through *subproperty* constraints).

Since the above holds for any pair of data nodes n_1, n_2 such that $n_1 \equiv_{\mathtt{fw}} n_2$ in $\mathtt{G}^{\infty}$, and for any of their $\mathtt{G}^{\infty}$ outgoing edges $n_1\ \mathtt{p}\ m_1$ and $n_2\ \mathtt{p}\ m_2$, hence $f(n_1) \equiv_{\mathtt{fw}} f(n_2)$ in $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$ holds.

Now, consider two data nodes $f(n_1), f(n_2)$ in $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$ such that $f(n_1) \equiv_{\mathtt{fw}} f(n_2)$ in $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$ and let us show that $n_1 \equiv_{\mathtt{fw}} n_2$ holds in $\mathtt{G}^{\infty}$.

If $f(n_1) \equiv_{\mathtt{fw}} f(n_2)$ holds in $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$, then for every triple $f(n_1)\ \mathtt{p}\ f(m_1)$ there exists a triple $f(n_2)\ \mathtt{p}\ f(m_2)$ such that $f(m_1) \equiv_{\mathtt{fw}} f(m_2)$ holds, and conversely for every triple $f(n_2)\ \mathtt{p}\ f(m_2)$ there exists a triple $f(n_1)\ \mathtt{p}\ f(m_1)$ such that $f(m_1) \equiv_{\mathtt{fw}} f(m_2)$ holds.

Let $\mathcal{P}^{\infty}_{f(n_1),f(n_2)\to f(m_1),f(m_2)}$ be the set of outgoing properties from $f(n_1)$ to $f(m_1)$ and from $f(n_2)$ to $f(m_2)$ in $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$.

In $\mathtt{G}_{/\equiv_{\mathtt{fw}}}$, the set of outgoing properties from $f(n_1)$ to $f(m_1)$, denoted $\mathcal{P}_{f(n_1)\to f(m_1)}$ is a subset of $\mathcal{P}^{\infty}_{fn_1),f(n_2)\to f(m_1),f(m_2)}$, since by definition the saturation of a graph only adds edges; similarly, in $\mathtt{G}_{/\equiv_{\mathtt{fw}}}$, the set of outgoing properties from $f(n_2)$ to $f(m_2)$, denoted $\mathcal{P}_{f(n_2)\to f(m_2)}$ is a subset of $\mathcal{P}^{\infty}_{f(n_1),f(n_2)\to f(m_1),f(m_2)}$, which may be different from $\mathcal{P}_{f(n_1)\to f(m_1)}$.

By definition of a $\equiv_{\mathtt{fw}}$-summary, the set of outgoing properties from n_1 to m_1 in $\mathtt{G}$ is exactly $\mathcal{P}_{f(n_1)\to f(m_1)}$ and similarly the set of outgoing properties from n_2 to m_2 in $\mathtt{G}$ is exactly $\mathcal{P}_{f(n_2)\to f(m_2)}$.

Since $\mathtt{G}$ and $\mathtt{G}_{/\equiv_{\mathtt{fw}}}$ have the same schema (Property 3), it follows that in $\mathtt{G}^{\infty}$, the set of outgoing properties from n_1 to m_1, and from n_2 to m_2, is exactly $\mathcal{P}^{\infty}_{f(n_1),f(n_2)\to f(m_1),f(m_2)}$ (data edges can only be added through *subproperty* constraints).

Since the above holds for any pair of data nodes $f(n_1), f(n_2)$ such that $f(n_1) \equiv_{\mathtt{fw}} f(n_2)$ in $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$, and for any of their $(\mathtt{G}_{/\equiv_{\mathtt{fw}}})^{\infty}$ outgoing edges $f(n_1)\ \mathtt{p}\ f(m_1)$ and $f(n_2)\ \mathtt{p}\ f(m_2)$, hence $n_1 \equiv_{\mathtt{fw}} n_2$ in $\mathtt{G}^{\infty}$ holds.

The proof for $\equiv_{\mathtt{bw}}$ directly derives from the above one by considering incoming edges instead of outgoing ones; the proof for $\equiv_{\mathtt{fb}}$ then derives from those of $\equiv_{\mathtt{fw}}$ and $\equiv_{\mathtt{bw}}$ by considering both incoming and outgoing edges. □

N Proof of Proposition 6

Proof. All our algorithms (global or incremental) start by identifying the class and property nodes: this is done retrieving all the subjects and objects of $\mathtt{S}_{\mathtt{G}}$ triples, and also all the objects of $\mathtt{T}_{\mathtt{G}}$ triples. As previously stated, triple stores routinely support such retrieval efficiently. Our algorithms start by representing these special schema nodes exactly by themselves, and copying in the summary all the schema triples from $\mathtt{S}_{\mathtt{G}}$. This exploits Proposition 3 ($\mathtt{G}$ and $\mathtt{G}_{\equiv}$ have the same schema triples).

Below, we show the correctness of **incremental** $\mathtt{W}$ **and** $\mathtt{S}$ **summarization** on $\mathtt{D}_{\mathtt{G}}$. The proof of Proposition 7 (below) extends this also to $\mathtt{T}_{\mathtt{G}}$ triples.

The correctness of **incremental** $\mathtt{W}$ **summarization** on $\mathtt{D}_{\mathtt{G}}$ follows from the fact that Algorithm **increm-**$\mathtt{W}$ preserves a set of invariants. Let $\mathtt{G}_k$ be the first k triples of $\mathtt{G}$, in the order in which they are traversed by

the algorithm. For any $1 \leq k \leq |\mathtt{G}|$, after applying **increm-**$\mathtt{W}$ on k data triples, the following invariants are preserved:

1. The source and target $src_{\mathtt{p}}$ and $trg_{\mathtt{p}}$ of any property $\mathtt{p}$ present in these k triples are known.
2. For any summarized triple $\mathtt{s}\ \mathtt{p}\ \mathtt{o}$, we have $f_{\mathtt{W}}(\mathtt{s}) = src_{\mathtt{p}}$ and $f_{\mathtt{W}}(\mathtt{o}) = trg_{\mathtt{p}}$; further, the summary contains the edge $f_{\mathtt{W}}(\mathtt{s})\ \mathtt{p}\ f_{\mathtt{W}}(\mathtt{o})$.

The preservation of these invariants is shown by considering all the cases which may occur for a given summarized triple $\mathtt{s}\ \mathtt{p}\ \mathtt{o}$: the subject $\mathtt{s}$ may have *already been seen* (in which case this triple may lead to a fusion), *or not* (in which case we create the new representative of $\mathtt{s}$), and similarly for $\mathtt{o}$. For $\mathtt{p}$ there are also two cases (depending on whether we had already encountered it or not, we may create $src_{\mathtt{p}}$ and $trg_{\mathtt{p}}$, or just fuse them with pre-existing representatives of $\mathtt{s}$ and $\mathtt{o}$). There are 8 cases overall. The replacements and fusions detailed in Table 5 guarantee these invariants.

While, for simplicity of presentation, Algorithm **increm-**$\mathtt{W}$ considers the possible fusions due to $\mathtt{s}$ and $\mathtt{o}$ separately, in reality, given that they may impact the same node(s) (e.g., if $f_{\mathtt{W}}(\mathtt{s}) = f_{\mathtt{W}}(\mathtt{o})$), all the replacements are first computed, then reconciled into a *list of summary node substitutions*, applied in all the data structures. For instance, suppose we need to replace summary node 3 with 1 because of a fusion on the subject side, and also summary node 5 with 3 because of a fusion on the object side. In this case, the algorithm will replace 5 and 3 directly with 1. If the replacements were applied sequentially, e.g., first 3 with 1, the second replacement would leave 3 (not 1) instead of 5, which would be an error.

Similarly, the correctness of **incremental** $\mathtt{S}$ **summarization** on $\mathtt{D}_{\mathtt{G}}$ follows from the fact that Algorithm **increm-**$\mathtt{S}$ preserves the following invariants after having been called on k successive data triples, with $1 \leq k \leq |\mathtt{G}|$:

1. The source and target clique $sc(\mathtt{p})$ and $tc(\mathtt{p})$ of any property $\mathtt{p}$ present in these k triples are known, and they contain $\mathtt{p}$.
2. For any summarized triple $\mathtt{s}\ \mathtt{p}\ \mathtt{o}$, we have $f_{\mathtt{S}}(\mathtt{s}) = sc(\mathtt{p})$ and $f_{\mathtt{S}}(\mathtt{o}) = tc(\mathtt{p})$; further, the summary contains the edge $f_{\mathtt{S}}(\mathtt{s})\ \mathtt{p}\ f_{\mathtt{S}}(\mathtt{o})$.
3. For any source clique sc and target clique tc of a node n appearing in the summarized triple, the summary contains exactly one node.
4. For any summary node m, the count $m_\sharp$ is exactly the cardinality of the set $\{n \in \mathtt{G} \mid f_{\mathtt{S}}(n) = m\}$.
5. For any summary edge $m \xrightarrow{\mathtt{p}} m'$, the count $e_\sharp$ is exactly the cardinality of the set $\{n \xrightarrow{\mathtt{p}} n'$ edge of $\mathtt{G} \parallel f_{\mathtt{S}}(n) = m$ and $f_{\mathtt{S}}(n') = m'\}$.
6. For any (subject, property) combination occurring in the summarized triples, the count $(\mathtt{sp})_\sharp$ is exactly the number of times this occurred in the triples. Similarly, fo r any (property, object) combination appearing in the summarized triples, the count $(\mathtt{po})_\sharp$ is exactly the number of times it appeared.

Like for **increm-**$\mathtt{W}$, there are eight cases depending on whether $\mathtt{s}$, $\mathtt{p}$ and $\mathtt{o}$ have been previously seen. Further, in the four cases where $\mathtt{s}$ has been seen, we may need to split $\mathtt{s}$'s representative, or not, and similarly for $\mathtt{o}$; thus, the six cases original cases where at least one of them had been seen lead to 12 cases (to which we add the remaining two, where neither $\mathtt{s}$ nor $\mathtt{o}$ had been seen), for a total of 14 cases.

Items 4, 5 and 6 are ensured during: the addition of an edge to the summary (this sets $e_\sharp$ to 1 or increases it); the assignment of representatives to nodes (this sets $m_\sharp$ to 1 or increments it); the edge repartition during split (this substracts from one edges $e_\sharp$ exactly the count that it adds to another new edge); and node replacements (which, when replacing u with v, either carry $u_\sharp$ into $v_\sharp$, if v did not exist

in the summary previously, or add $u_\sharp$ to $v_\sharp$ if it did). Together, 4, 5 and 6 ensure the correctness of the **split** algorithm (explained in Section 7.1).

The previous items are ensured by the creation of summary nodes (at most one exists at any time for a given source and target clique), fusing cliques (this guarantees each property is in the right clique, and remove cliques input to the fusion), and replacing / fusing summary nodes, as well as from the correctness of the split procedure. □

O Proof of Proposition 7

Proof. First, recall that TW and TS summarization start with the type triples, which means all type nodes are detected and represented according to their class sets, before the data triples are summarized. This entails that among the cases which occur for W and S summarization (8, respectively, 14, see discussion in the proof of Proposition 6), those in which the subject, respectively, the object was already represented are further divided in two, depending on whether the subject, respectively, object was a typed node.

This shows that *incremental* TW *summarization handles a superset of the cases handled by the* W *one*, and similarly for TS and TS. Thus, **increm-**TW, respectively, **increm-**TS preserve all the invariants of **increm-**W, respectively, **increm-**S[8], with some additions, which we highlight in italics below:

- Additions of TW summarization w.r.t. W:
  1. The source and target $src_\mathtt{p}$ and $trg_\mathtt{p}$ of any property p present *with an untyped source, respectively, an untyped target* in the summarized triples are known.
  2. For any summarized triple s p o, we have $f_\mathtt{W}(\mathtt{s}) = src_\mathtt{p}$ *if* s *is untyped* and $f_\mathtt{W}(\mathtt{o}) = trg_\mathtt{p}$ *if* o *is untyped*; further, the summary contains the edge $f_\mathtt{W}(\mathtt{s})$ p $f_\mathtt{W}(\mathtt{o})$.
- Additions of TS summarization w.r.t. S:
  1. The source and target clique $sc(\mathtt{p})$ and $tc(\mathtt{p})$ of any property p present in these k triples *with an untyped source, respectively, with an untyped target* are known, and they contain p.
  2. For any summarized triple s p o, we have $f_\mathtt{S}(\mathtt{s}) = sc(\mathtt{p})$ *if* s *is untyped*, and $f_\mathtt{S}(\mathtt{o}) = tc(\mathtt{p})$ *if* o *is untyped*; further, the summary contains the edge $f_\mathtt{S}(\mathtt{s})$ p $f_\mathtt{S}(\mathtt{o})$.

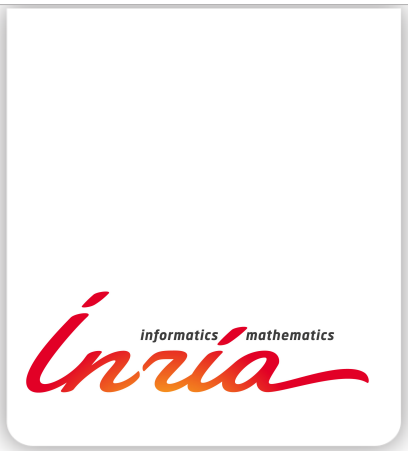
Further, they also preserve:

7. The summary contains one node for each set of classes belonging to some resource in the input.
8. For any node n with a non empty class set, $f_\mathtt{TW}(n)$ (respectively, $f_\mathtt{TS}(n)$) is the nodes corresponding to the class set of n.

These invariants are ensured by the way in which we collect all class sets during the initial traversal of type triples (common to the TW and TS algorithms). Further, during the TW and TS summarization, as said in Section 7.2, the representatives of typed nodes never fuse, and never split.

The 6 invariants from the above proof of Proposition 6 ensure the correct summarization of $\mathtt{D}_\mathtt{G}$ triples when s and o are untyped. Together with the two above, they also ensure the correct summarization of triples having a typed source and/or object. □

[8] Note that in the particular case of triples connecting untyped nodes, the algorithms coincide.



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399