SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples*

Aidan Hogan[1], Jeff Z. Pan[2], Axel Polleres[1], and Stefan Decker[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
{*firstname.lastname*}@deri.org

[2] Dpt. of Computing Science, University of Aberdeen
jeff.z.pan@abdn.ac.uk

Abstract. In this paper, we discuss optimisations of rule-based materialisation approaches for reasoning over large static RDF datasets. We generalise and re-formalise what we call the "partial-indexing" approach to scalable rule-based materialisation: the approach is based on a separation of terminological data, which has been shown in previous and related works to enable highly scalable and distributable reasoning for specific rulesets; in so doing, we provide some completeness propositions with respect to semi-naïve evaluation. We then show how related work on template rules – T-Box-specific dynamic rulesets created by binding the terminological patterns in the static ruleset – can be incorporated and optimised for the partial-indexing approach. We evaluate our methods using LUBM(10) for RDFS, pD* (OWL Horst) and OWL 2 RL, and thereafter demonstrate pragmatic distributed reasoning over 1.12 billion Linked Data statements for a subset of OWL 2 RL/RDF rules we argue to be suitable for Web reasoning.

1 Introduction

More and more structured data is being published on the Web in conformance with the Resource Description Framework (RDF) for disseminating machine-readable information, forming what is often referred to as the "Web of Data". This data is no longer purely academic: in particular, the Linked Data community – by promoting pragmatic best-practices and applications – has overseen RDF exports from, for example, corporate bodies (e.g., BBC, New York Times, Freebase), community driven efforts (e.g., Wikipedia, GeoNames), the biomedical domain (e.g., DrugBank, Linked Clinical Trials) and governmental bodies (e.g., data.gov, data.gov.uk). At a conservative estimate, there now exists tens of billions of RDF statements on the Web.

Sitting atop RDF are the RDF Schema (RDFS) and Web Ontology Language (OWL) standards. Primarily, RDFS and OWL allow for defining the relationships between the classes and properties used to organise and describe entities, providing a declarative and extensible domain of discourse through use of rich formal semantics. One could thereafter view the Web of Data as a massive, heterogeneous, collaboratively edited knowledge-base amenable for reasoning: however, the prospect of applying reasoning over (even subsets of) the Web of Data raises unique challenges, the most obvious of which are the need for scale, and tolerance to noisy, conflicting and impudent data [6].

* The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), by the EU MOST project, the EPSRC LITRO project, and by an IRCSET Scholarship.

Inspired by requirements for the Semantic Web Search Engine (SWSE) project [9] – which aims to offer search and browsing over Linked Data – in previous work we investigated pragmatic and scalable reasoning for Web data through work on the Scalable Authoritative OWL Reasoner (SAOR) [7,8]; we discussed the formulation and suitability of a set of rules inspired by pD* [16] for materialisation over Web data. We gave particular focus to scalability and Web tolerance showing that by abandoning completeness, materialisation over a diverse Web dataset – in the order of a billion statements – is entirely feasible wrt. a significant fragment of OWL semantics. From the scalability perspective, we introduced a partial-indexing approach based on a separation of terminological data from assertional data in our rule execution model: terminological data – the most frequently accessed segment of the knowledge base for reasoning which in our scenario represents only a small fraction of the overall data [8] – is stored and indexed in-memory for fast access, whereas the bulk of (assertional) data is processed by file-scans. Related approaches have since appeared in the literature which use a separation of terminological data for applying distributed RDFS and pD* reasoning over datasets containing hundreds of millions, billions and hundreds of billions of statements [19,18, 17]. However, each of these approaches has discussed completeness and implementation/optimisation based on the specific ruleset at hand.

In this paper, we reformulate the partial-indexing approach – generalising to arbitrary rulesets – and discuss when it is (i) complete with respect to standard rule closure; and (ii) appropriate and scalable. We then introduce generic optimisations based on "template rules" – where terminological data is bound by the rules prior to accessing the A-Box – and provide some initial evaluation over a small LUBM dataset for RDFS, pD*, and OWL 2 RL/RDF. Thereafter, we look to apply our optimisations for scalable and distributed Linked Data reasoning, initially reintroducing our *authoritative reasoning* algorithm which incorporates provenance, detailing distribution of our approach, and then providing evaluation for reasoning over 1.12b Web triples.

2 Preliminaries

Before we continue, we briefly introduce some concepts prevalent throughout the paper. We use notation and nomenclature as is popular in the literature (cf. [4,8]). Herein, we denote infinite sets by $\mathbf{S}$ and corresponding finite subsets by $\mathcal{S}$.

2.1 RDF and Rules

RDF Constant Given the set of URI references $\mathbf{U}$, the set of blank nodes $\mathbf{B}$, and the set of literals $\mathbf{L}$, the set of *RDF constants* is denoted by $\mathbf{C} := \mathbf{U} \cup \mathbf{B} \cup \mathbf{L}$.

RDF Triple A triple $t := (s, p, o) \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times \mathbf{C}$ is called an *RDF triple*, where *s* is called subject, *p* predicate, and *o* object. A triple $t := (s, p, o) \in \mathbf{G}, \mathbf{G} := \mathbf{C} \times \mathbf{C} \times \mathbf{C}$ is called a *generalised triple*, which allows any RDF constant in any triple position: hence forth, we assume generalised triples [2]. We call a finite set of triples $\mathcal{G} \subset \mathbf{G}$ a *graph*. (For brevity, we sometimes use `r:` for the RDFS namespace, `o:` for OWL namespace, and `f:` for the well-known FOAF namespace; we use '`a`' as a shortcut for `rdf:type`.)

Triple Pattern, Basic Graph Pattern A *triple pattern* is a generalised triple where variables from the set $\mathbf{V}$ are allowed; i.e.: $t^v := (s^v, p^v, o^v) \in \mathbf{G^V}, \mathbf{G^V} := (\mathbf{C} \cup \mathbf{V}) \times (\mathbf{C} \cup \mathbf{V}) \times (\mathbf{C} \cup \mathbf{V})$. We call a set (to be read as conjunction) of triple patterns $\mathcal{G}^\mathbf{V} \subset \mathbf{G^V}$ a *basic graph pattern*. We denote the set of variables in graph pattern $\mathcal{G}^\mathbf{V}$ by $\mathbf{V}(\mathcal{G}^\mathbf{V})$.

Variable Bindings Let $\mathbf{M}$ be the set of endomorphic *variable binding mappings* $\mathbf{V} \cup \mathbf{C} \rightarrow \mathbf{V} \cup \mathbf{C}$ which map every constant $c \in \mathbf{C}$ to itself and every variable $v \in \mathbf{V}$ to an element of the set $\mathbf{C} \cup \mathbf{V}$. A triple t is a *binding* of a triple pattern $t^v := (s^v, p^v, o^v)$ iff there exists $\mu \in \mathbf{M}$, such that $t = \mu(t^v) = (\mu(s^v), \mu(p^v), \mu(o^v))$. A graph $\mathcal{G}$ is a binding of a graph pattern $\mathcal{G}^{\mathbf{V}}$ iff there exists a mapping $\mu \in \mathbf{M}$ such that $\bigcup_{t^v \in \mathcal{G}^{\mathbf{V}}} \mu(t^v) = \mathcal{G}$; we use the shorthand $\mu(\mathcal{G}^{\mathbf{V}}) = \mathcal{G}$. We use $\mathbf{M}(\mathcal{G}^{\mathbf{V}}, \mathcal{G}) := \{\mu \mid \mu(\mathcal{G}^{\mathbf{V}}) \subseteq \mathcal{G}, \mu(v) = v \text{ if } v \notin \mathbf{V}(\mathcal{G}^{\mathbf{V}})\}$ to denote the set of variable binding mappings for graph pattern $\mathcal{G}^{\mathbf{V}}$ in graph $\mathcal{G}$ which map variables outside $\mathcal{G}^{\mathbf{V}}$ to themselves.

Inference Rule We define an *inference rule* r as the pair $(\mathcal{A}nte_r, \mathcal{C}on_r)$, where the *antecedent* (or *body*) $\mathcal{A}nte_r \subset \mathbf{G}^{\mathbf{V}}$ and the *consequent* (or *head*) $\mathcal{C}on_r \subset \mathbf{G}^{\mathbf{V}}$ are basic graph patterns such that $\mathbf{V}(\mathcal{C}on_r) \subseteq \mathbf{V}(\mathcal{A}nte_r)$ (range restricted) – rules with empty antecedents model axiomatic triples. We write inference rules as $\mathcal{A}nte_r \Rightarrow \mathcal{C}on_r$.

Rule Application and Standard Closure A *rule application* is the immediate consequences $T_r(\mathcal{G}) := \bigcup_{\mu \in \mathbf{M}(\mathcal{A}nte_r, \mathcal{G})}(\mu(\mathcal{C}on_r) \setminus \mu(\mathcal{A}nte_r))$ of a rule r on a graph $\mathcal{G}$; accordingly, for a ruleset $\mathcal{R}$, $T_{\mathcal{R}}(\mathcal{G}) := \bigcup_{r \in \mathcal{R}} T_r(\mathcal{G})$. Now, let $\mathcal{G}_{i+1} := \mathcal{G}_i \cup T_{\mathcal{R}}(\mathcal{G}_i)$ and $\mathcal{G}_0 := \mathcal{G}$; the *exhaustive application* of the $T_{\mathcal{R}}$ operator on a graph $\mathcal{G}$ is then the least fixpoint (the smallest value for n) such that $\mathcal{G}_n = T_{\mathcal{R}}(\mathcal{G}_n)$. We call $\mathcal{G}_n$ the *closure* of $\mathcal{G}$ wrt. ruleset $\mathcal{R}$, denoted as $Cl_{\mathcal{R}}(\mathcal{G})$, or succinctly $\overline{\mathcal{G}}$ where the ruleset is obvious.

The above closure takes a graph and a ruleset and recursively applies the rules over the union of the original graph and the inferences until a fixpoint. Usually, this would consist of indexing all input and inferred triples; however, the cost of indexing and performing query-processing over large graphs can become prohibitively expensive. Thus, in [7] we originally proposed an alternate method based on a separation of terminological data, which we now generalise and discuss.

3 Partial Indexing Approach: Separating Terminological Data

In the field of Logic Programming, the notion of a 'linear program' refers loosely to a ruleset where only one pattern in each rule is recursive [12]. Our partial indexing approach is optimised for linear rules, where the non-recursive segment of the data is identified, separated and prepared, and thereafter each recursive pattern can then be bound via a triple-by-triple stream: we cater for non-linear rules, but as the number of recursive rules, the amount of recursion, and the amount of recursive data involved increases, our approach performs worse than the "full-indexing" approach.

Specifically regarding RDFS and OWL, the terminological segment of the data presents itself as relatively small and 'non-recursive' (or at least, mostly only recursive within itself), which can be leveraged for partial indexing. Herein, we define our notion of RDF(S)/OWL terminological data. (To generalise the following, the reader can consider terminological data as the RDFS/OWL archetype for any non-recursive and sufficiently small element of the data commonly required during rule application.)

Meta-class We consider a *meta-class* as a class specifically of classes or properties; i.e., the members of a meta-class are themselves either classes or properties. Herein, we restrict our notion of meta-classes to the set defined in RDF(S) and OWL specifications, where examples include `rdf:Property`, `rdfs:Class`, `owl:Restriction`, `owl:DatatypeProperty`, `owl:TransitiveProperty`, etc.; `rdfs:Resource`, `rdfs:Literal`, e.g., are not meta-classes.

Meta-property A *meta-property* is one which has a meta-class as its domain; again, we restrict our notion of meta-properties to the set defined in RDF(S) and OWL specifications, where examples include `rdfs:domain`, `rdfs:subClassOf`, `owl:hasKey`, `owl:inverseOf`, `owl:oneOf`, `owl:onProperty`, `owl:unionOf`, etc.; `rdf:type`, `owl:sameAs`, `rdfs:label`, e.g., do *not* have a meta-class as domain.

Terminological Triple We define the set of *terminological triples* $\mathbf{T} \subset \mathbf{G}$ as the union of (i) triples with `rdf:type` as predicate and a meta-class as object; (ii) triples with a meta-property as predicate; (iii) triples forming a *valid* RDF list whose head is the object of a meta-property (e.g., a list used for `owl:unionOf`, etc.).

Terminological/Assertional Pattern We refer to a *terminological -triple/-graph pattern* as one whose instance can only be a terminological triple or, resp., a set thereof. An *assertional pattern* is any pattern which is not terminological.

Given the above notions of terminological data/patterns, we now define a $\mathcal{T}$-split inference rule where part of the rule body is strictly matched by terminological data.

Definition 1. $\mathcal{T}$-split inference rule *Given a rule $r := (\mathit{Ante}_r, \mathit{Con}_r)$, we define a* $\mathcal{T}$-split rule r^τ *as the triple $(\mathit{Ante}^{\mathcal{T}}_{r^\tau}, \mathit{Ante}^{\mathcal{G}}_{r^\tau}, \mathit{Con})$ where $\mathit{Ante}^{\mathcal{T}}_{r^\tau}$ is the set of terminological patterns in Ante_r, and $\mathit{Ante}^{\mathcal{G}}_{r^\tau} := \mathit{Ante}_r \setminus \mathit{Ante}^{\mathcal{T}}_{r^\tau}$. We denote the set of all* $\mathcal{T}$-split rules *by $\mathbf{R}^\tau$, and the mapping of a rule to its $\mathcal{T}$-split version as $\tau : \mathbf{R} \to \mathbf{R}^\tau; r \mapsto r^\tau$. We additionally give the convenient sets $\mathbf{R}^\emptyset := \{r^\tau \mid \mathit{Ante}^{\mathcal{T}}_{r^\tau} = \emptyset, \mathit{Ante}^{\mathcal{G}}_{r^\tau} = \emptyset\}$, $\mathbf{R}^{\mathbf{T}\emptyset} := \{r^\tau \mid \mathit{Ante}^{\mathcal{T}}_{r^\tau} \neq \emptyset, \mathit{Ante}^{\mathcal{G}}_{r^\tau} = \emptyset\}$, $\mathbf{R}^{\emptyset\mathbf{G}} := \{r^\tau \mid \mathit{Ante}^{\mathcal{T}}_{r^\tau} = \emptyset, \mathit{Ante}^{\mathcal{G}}_{r^\tau} \neq \emptyset\}$, $\mathbf{R}^{\mathbf{TG}} := \{r^\tau \mid \mathit{Ante}^{\mathcal{T}}_{r^\tau} \neq \emptyset, \mathit{Ante}^{\mathcal{G}}_{r^\tau} \neq \emptyset\}$, $\mathbf{R}^{\mathbf{G}} := \mathbf{R}^{\mathbf{TG}} \cup \mathbf{R}^{\emptyset\mathbf{G}}$ and $\mathbf{R}^{\mathbf{T}} := \mathbf{R}^{\mathbf{T}\emptyset} \cup \mathbf{R}^{\mathbf{TG}}$ as the set of all $\mathcal{T}$-split rules with an empty antecedent, only terminological patterns, only assertional patterns, both types of patterns, some terminological patterns, and some assertional pattern respectively, where $\mathbf{R}^\tau = \mathbf{R}^\emptyset \cup \mathbf{R}^{\mathbf{T}\emptyset} \cup \mathbf{R}^{\emptyset\mathbf{G}} \cup \mathbf{R}^{\mathbf{TG}} = \mathbf{R}^\emptyset \cup \mathbf{R}^{\mathbf{T}} \cup \mathbf{R}^{\mathbf{G}}$. We also give the sets $\mathbf{R}^{\mathbf{G^1}} := \{r^\tau \in \mathbf{R}^{\mathbf{G}} : |\mathit{Ante}^{\mathcal{G}}_{r^\tau}| = 1\}$, $\mathbf{R}^{\mathbf{G^n}} := \{r^\tau \in \mathbf{R}^{\mathbf{G}} : |\mathit{Ante}^{\mathcal{G}}_{r^\tau}| > 1\}$, denoting the set of linear and non-linear rules respectively. Given a $\mathcal{T}$-split ruleset $\mathcal{R}^\tau$, herein we may use, e.g., $\mathcal{R}^{\mathbf{G}}$ to denote $\mathcal{R}^\tau \cap \mathbf{R}^{\mathbf{G}}$.*

Example 1. For the rule $r :=$ `(?c1,r:subClassOf,?c2)` $\wedge$ `(?x,a,?c1)` $\Rightarrow$ `(?x,a,?c2)`, $\mathit{Ante}^{\mathcal{T}} := \{$`(?c1,r:subClassOf,?c2)`$\}$ and $\mathit{Ante}^{\mathcal{G}} := \{$`(?x,a,?c1)`$\}$. Underlining $\mathit{Ante}^{\mathcal{T}}$, we write $\tau(r) := r^\tau := \underline{\texttt{(?c1,r:subClassOf,?c2)}} \wedge$ `(?x,a,?c1)` $\Rightarrow$ `(?x,a,?c2)`.

We then define our T-Box as the set of terminological triples in a given graph which are required by the terminological patterns of a given ruleset.

Definition 2. T-Box/A-Box *Given a graph $\mathcal{G}$ and a $\mathcal{T}$-split ruleset $\mathcal{R}^\tau$, let $\mathcal{R}^{\mathbf{T}} := \mathcal{R}^\tau \cap \mathbf{R}^{\mathbf{T}}$ represent the subset of rules in $\mathcal{R}^\tau$ which require terminological data; the T-Box of $\mathcal{G}$ wrt. $\mathcal{R}^\tau$ is then $\mathbf{T}(\mathcal{G}, \mathcal{R}^\tau) := \bigcup_{r^\tau \in \mathcal{R}^{\mathbf{T}}} \bigcup_{t^v \in \mathit{Ante}^{\mathcal{T}}_{r^\tau}} \bigcup_{\mu \in \mathbf{M}(\{t^v\}, \mathcal{G})} \mu(t^v)$, representing the subset of terminological triples in $\mathcal{G}$ which satisfy a terminological pattern of a rule antecedent ($\mathit{Ante}^{\mathcal{T}}_{r^\tau}$) in $\mathcal{R}^\tau$; where ruleset and graph are obvious, we may abbreviate $\mathbf{T}(\mathcal{G}, \mathcal{R}^\tau)$ to simply $\mathcal{T}$. Our A-Box is synonymous with $\mathcal{G}$: i.e., we also consider our T-Box as part of our A-Box in a form of unidirectional meta-modelling.*

Given the notion of a $\mathcal{T}$-split rule and our T-Box, we can now define how $\mathcal{T}$-split rules are applied, and how $\mathcal{T}$-split closure is achieved wrt. a static T-Box.

Definition 3. **$\mathcal{T}$-split rule application and closure** *We define a $\mathcal{T}$-*split rule application *for a $\mathcal{T}$-split rule r^{τ} wrt. a graph $\mathcal{G}$ to be:*

$$T_{r^\tau}(\mathcal{T},\mathcal{G}) := \bigcup_{\mu_0 \in \mathbf{M}(\mathcal{A}nte^{\mathcal{T}}_{r^\tau},\mathcal{T})} \; \bigcup_{\mu_1 \in \mathbf{M}(\mu_0(\mathcal{A}nte^{\mathcal{G}}_{r^\tau}),\mathcal{G})} (\mu_0 \circ \mu_1)(\mathcal{C}on_{r^\tau}) \tag{1}$$

here formalising the notion that the terminological patterns of the rule are strictly instantiated from a separate T-Box $\mathcal{T}$. Again, for a $\mathcal{T}$-split ruleset $\mathcal{R}^{\tau}$, $T_{\mathcal{R}^\tau}(\mathcal{T},\mathcal{G}) := \bigcup_{r^\tau \in \mathcal{R}^\tau} T_{r^\tau}(\mathcal{T},\mathcal{G})$. Now, let $\mathcal{A}x$ denote the set of axiomatic triples given by $\mathcal{R}^{\tau}$ (the same set as for $\mathcal{R}$), and $\mathcal{T}_0 := \mathbf{T}(\mathcal{G} \cup \mathcal{A}x, \mathcal{R}^\tau)$ be our initial T-Box derived from $\mathcal{G}$ and axiomatic triples, and $\mathcal{T}_{i+1} := \mathcal{T}_i \cup \mathbf{T}(T_{\mathcal{R}^{\mathbf{T}\emptyset}}(\mathcal{T}_i,\emptyset),\mathcal{R}^\tau)$; we define our closed T-Box *as $\mathcal{T}_n$ for the least value of n such that $\mathcal{T}_n = \mathcal{T}_n \cup T_{\mathcal{R}^{\mathbf{T}\emptyset}}(\mathcal{T}_n,\emptyset)$, denoted $\overline{\mathcal{T}}^{\tau}$, representing the closure of our initial T-Box wrt. rules requiring only terminological knowledge. Finally, let $\mathcal{G}^{\tau}_0 := \mathcal{G} \cup \overline{\mathcal{T}}^{\tau} \cup \mathcal{A}x$ and $\mathcal{G}^{\tau}_{i+1} := \mathcal{G}^{\tau}_i \cup T_{\mathcal{R}^{\mathbf{G}}}(\overline{\mathcal{T}}^{\tau},\mathcal{G}^{\tau}_i)$; we now define the exhaustive application of the $T_{\mathcal{R}^\tau}$ operator on a graph $\mathcal{G}$ wrt. a static T-Box $\mathcal{T}$ as being upto the least fixpoint such that $\mathcal{G}^{\tau}_n = T_{\mathcal{R}^{\mathbf{G}}}(\overline{\mathcal{T}}^{\tau},\mathcal{G}^{\tau}_n)$. We call $\mathcal{G}^{\tau}_n$ the $\mathcal{T}$-split closure of $\mathcal{G}$ with respect to the $\mathcal{T}$-split ruleset $\mathcal{R}^\tau$, denoted as $Cl_{\mathcal{R}^\tau}(\mathcal{T},\mathcal{G})$ or simply $\overline{\mathcal{G}}^{\tau}$.*

The $\mathcal{T}$-split closure algorithm consists of two main steps: (i) deriving the closed T-Box from axiomatic triples, the input graph, and recursively applied $\mathbf{R}^{\mathbf{T}\emptyset}$ rules; (ii) applying 'A-Box' reasoning for all triples wrt. the $\mathbf{R}^{\mathbf{G}}$ rules and the static T-Box. We now give some propositions relating the $\mathcal{T}$-split closure with the standard rule application closure described in the preliminaries; firstly, we must give an auxiliary proposition which demonstrates how mappings for sub-graphs-patterns can be combined to give the mappings for the entire graph pattern, which relates to the $\mathcal{T}$-split rule application.

Proposition 1. *For any graph $\mathcal{G}$ and graph pattern $\mathcal{G}^{\mathbf{V}} := \mathcal{G}^{\mathbf{V}}_a \cup \mathcal{G}^{\mathbf{V}}_b$, it holds that $\mathbf{M}(\mathcal{G}^{\mathbf{V}},\mathcal{G}) = \{\mu_b \circ \mu_a \mid \mu_a \in \mathbf{M}(\mathcal{G}^{\mathbf{V}}_a,\mathcal{G}), \mu_b \in \mathbf{M}(\mu_a(\mathcal{G}^{\mathbf{V}}_b),\mathcal{G})\}$.*

Proof. Firstly, $\mu_b \circ \mu_a \in \mathbf{M}$ since μ_a and μ_b are endomorphic. By definition, $(\mu_b \circ \mu_a)(c) = c$ for $c \in \mathbf{C}$. Next, we need to show that $(\mu_a \circ \mu_b)(v) = v$ if $v \notin \mathbf{V}(\mathcal{G}^{\mathbf{V}})$: since by definition $\mu_a(v) = v$ if $v \notin \mathcal{G}^{\mathbf{V}}_a$ and $\mu_b(v) = v$ if $v \notin \mu_a(\mathcal{G}^{\mathbf{V}}_b)$, and since $\mathbf{V}(\mu_a(\mathcal{G}^{\mathbf{V}}_b)) \subseteq \mathbf{V}(\mathcal{G}^{\mathbf{V}}_b)$ and $\mathbf{V}(\mathcal{G}^{\mathbf{V}}) = \mathbf{V}(\mathcal{G}^{\mathbf{V}}_a) \cup \mathbf{V}(\mathcal{G}^{\mathbf{V}}_b)$, then $(\mu_b \circ \mu_a)(v) = v$ if $v \notin \mathbf{V}(\mathcal{G}^{\mathbf{V}})$. By definition, $\mu_a(\mathcal{G}^{\mathbf{V}}_a) \subseteq \mathcal{G}$ and thus we have $\mathbf{V}(\mu_a(\mathcal{G}^{\mathbf{V}}_a)) = \emptyset$, and $\mu_a(\mathcal{G}^{\mathbf{V}}_a) = (\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}_a)$; again by definition we have $(\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}_b) \subseteq \mathcal{G}$, and so $(\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}_a) \cup (\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}_b) = (\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}_a \cup \mathcal{G}^{\mathbf{V}}_b) = (\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}) \subseteq \mathcal{G}$. We now have $\mu_b \circ \mu_a \in \mathbf{M}(\mathcal{G}^{\mathbf{V}},\mathcal{G})$ for every $\mu_a \in \mathbf{M}(\mathcal{G}^{\mathbf{V}}_a,\mathcal{G}), \mu_b \in \mathbf{M}(\mu_a(\mathcal{G}^{\mathbf{V}}_b),\mathcal{G})$, and need to show that for every $\mu \in \mathbf{M}(\mathcal{G}^{\mathbf{V}},\mathcal{G})$, there exists a $(\mu_b \circ \mu_a)$ such that $(\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}) = \mu(\mathcal{G}^{\mathbf{V}})$; by definition, we know that there exists a μ_a such that $\mu_a(\mathcal{G}^{\mathbf{V}}_a) = \mu(\mathcal{G}^{\mathbf{V}}_a)$ for any μ as defined, and that for every such μ_a there exists a μ_b such that $(\mu_b \circ \mu_a)(\mathcal{G}^{\mathbf{V}}_b) = (\mu \circ \mu_a)(\mathcal{G}^{\mathbf{V}}_b) = \mu(\mathcal{G}^{\mathbf{V}}_b)$, and hence the proposition holds. □

Theorem 1. **Soundness** *For any given ruleset $\mathcal{R} \subset \mathbf{R}$, its $\mathcal{T}$-split version $\mathcal{R}^\tau := \tau(\mathcal{R})$, and any graph $\mathcal{G}$, it holds that $\overline{\mathcal{G}}^{\tau} \subseteq \overline{\mathcal{G}}$.*

Proof. Clearly, $\mathcal{A}x$ gives the same set of triples for $\mathcal{R}^\tau$ and $\mathcal{R}$, and thus $\mathcal{T}_0 \subseteq \overline{\mathcal{G}}$ since $\mathbf{T}(\mathcal{G} \cup \mathcal{A}x, \mathcal{R}^\tau) \subseteq \mathcal{G} \cup \mathcal{A}x \subseteq \overline{\mathcal{G}}$. From Proposition 1, it follows that $\mathbf{M}(\mathcal{A}nte_r,\mathcal{G}) = \mathbf{M}(\mathcal{A}nte^{\mathcal{T}}_{r^\tau} \cup \mathcal{A}nte^{\mathcal{G}}_{r^\tau},\mathcal{G}) = \{\mu_0 \circ \mu_1 \mid \mu_0 \in \mathbf{M}(\mathcal{A}nte^{\mathcal{T}}_{r^\tau},\mathcal{G}), \mu_1 \in \mathbf{M}(\mu_0(\mathcal{A}nte^{\mathcal{G}}_{r^\tau}),\mathcal{G})\}$; we can then show that $T_r(\mathcal{G}) = T_{r^\tau}(\mathcal{G},\mathcal{G})$ by replacing $\mathcal{T}$ with $\mathcal{G}$ in Equation 1, from which follows $T_{\mathcal{R}}(\mathcal{G}) = T_{\mathcal{R}^\tau}(\mathcal{G},\mathcal{G})$. Given that $T_{\mathcal{R}^\tau_a}(\mathcal{G},\mathcal{G}) \subseteq T_{\mathcal{R}^\tau}(\mathcal{G},\mathcal{G})$ if $\mathcal{R}^\tau_a \subseteq \mathcal{R}^\tau$, and $T_{\mathcal{R}^\tau}(\mathcal{G}_a, \mathcal{G}_b) \subseteq T_{\mathcal{R}^\tau}(\mathcal{G}, \mathcal{G})$ if $\mathcal{G}_a \subseteq \mathcal{G}$ and $\mathcal{G}_b \subseteq \mathcal{G}$ – i.e., that our rule applications are monotonic – we can show by induction that $\overline{\mathcal{T}}^\tau \subseteq \overline{\mathcal{G}}$: given $\mathcal{T}_0 \subseteq \overline{\mathcal{G}}$ from above, we can say that $\mathcal{T}_{i+1} \subseteq \overline{\mathcal{G}}$ iff $\mathcal{T}_i \subseteq \overline{\mathcal{G}}$ since $\mathbf{T}(T_{\mathcal{R}^{\mathbf{T}\emptyset}}(\mathcal{T}_i, \emptyset)) \subseteq T_{\mathcal{R}^{\mathbf{T}\emptyset}}(\mathcal{T}_i, \mathcal{T}_i) \subseteq T_{\mathcal{R}}(\mathcal{T}_i) \subseteq \overline{\mathcal{G}}$. Now, clearly $\mathcal{G}_0^\tau \subseteq \overline{\mathcal{G}}$, and since $T_{\mathcal{R}^{\mathbf{G}}}(\overline{\mathcal{T}}^\tau, \mathcal{G}_i^\tau) \subseteq T_{\mathcal{R}^\tau}(\mathcal{G}_i^\tau, \mathcal{G}_i^\tau) = T_{\mathcal{R}}(\mathcal{G}_i^\tau) \subseteq \overline{\mathcal{G}}$, we can say that if $\mathcal{G}_i^\tau \subseteq \overline{\mathcal{G}}$, then $\mathcal{G}_{i+1}^\tau \subseteq \overline{\mathcal{G}}$; by induction, $\overline{\mathcal{G}}^\tau \subseteq \overline{\mathcal{G}}$. □

Theorem 2. Conditional Completeness *If* $\overline{\mathcal{T}}^\tau = \mathbf{T}(\overline{\mathcal{G}}^\tau, \mathcal{R}^\tau)$, *then* $\overline{\mathcal{G}}^\tau = \overline{\mathcal{G}}$.

Proof. First, $T_{\mathcal{R}^\tau}(\mathbf{T}(\mathcal{G}, \mathcal{R}^\tau), \mathcal{G}) = T_{\mathcal{R}^\tau}(\mathcal{G}, \mathcal{G})$ since by definition $\mathbf{T}(\mathcal{G}, \mathcal{R}^\tau)$ only removes triples from $\mathcal{G}$ that cannot be bound by terminological patterns in $\mathcal{R}^\tau$. Given the criteria $\overline{\mathcal{G}}^\tau = \overline{\mathcal{G}}^\tau \cup T_{\mathcal{R}^{\mathbf{G}}}(\overline{\mathcal{T}}^\tau, \overline{\mathcal{G}}^\tau)$ – or, rephrasing, $T_{\mathcal{R}^{\mathbf{G}}}(\overline{\mathcal{T}}^\tau, \overline{\mathcal{G}}^\tau) \subseteq \overline{\mathcal{G}}^\tau$ – we first know that $\mathcal{A}x \cup \overline{\mathcal{T}}^\tau \cup \mathcal{G} = \mathcal{G}_0^\tau \subseteq \overline{\mathcal{G}}^\tau$. Thus, $T_{\mathcal{R}^\tau}(\overline{\mathcal{T}}^\tau, \overline{\mathcal{G}}^\tau) = T_{\mathcal{R}^{\mathbf{G}}}(\overline{\mathcal{T}}^\tau, \overline{\mathcal{G}}^\tau) \subseteq \overline{\mathcal{G}}^\tau$. If $\overline{\mathcal{T}}^\tau = \mathbf{T}(\overline{\mathcal{G}}^\tau, \mathcal{R}^\tau)$, then $T_{\mathcal{R}^\tau}(\overline{\mathcal{T}}^\tau, \overline{\mathcal{G}}^\tau) = T_{\mathcal{R}^\tau}(\mathbf{T}(\overline{\mathcal{G}}^\tau, \mathcal{R}^\tau), \overline{\mathcal{G}}^\tau) = T_{\mathcal{R}^\tau}(\overline{\mathcal{G}}^\tau, \overline{\mathcal{G}}^\tau) = T_{\mathcal{R}}(\overline{\mathcal{G}}^\tau)$, which gives $\mathcal{G}_0 \subseteq \overline{\mathcal{G}}^\tau \subseteq \overline{\mathcal{G}}$: i.e., $\overline{\mathcal{G}}^\tau$ is known to be the partial closure of $\mathcal{G}$. Given the fixpoint condition $\overline{\mathcal{G}} = \overline{\mathcal{G}} \cup T_{\mathcal{R}}(\overline{\mathcal{G}})$, then $\overline{\mathcal{G}}^\tau$ must be the fixpoint: $\overline{\mathcal{G}}^\tau = \overline{\mathcal{G}}$. □

Proposition 2. *A triple* $t \in \mathbf{T}(\overline{\mathcal{G}}^\tau, \mathcal{R}^\tau) \setminus \overline{\mathcal{T}}^\tau$ *can only be produced for* $\overline{\mathcal{G}}^\tau$ *through an inference for a rule in* $\mathbf{R}^{\mathbf{G}}$.

Proof. Any T-Box triples in the original graph, or T-Box triples produced by the 'closure' of $\mathbf{R}^\emptyset$ rules are added to the initial T-Box $\mathcal{T}_0$. Any T-Box triples produced by the closure of $\mathbf{R}^{\mathbf{T}\emptyset}$ rules over $\mathcal{T}_0$ are added to the closed T-Box $\overline{\mathcal{T}}^\tau$. Since $\mathbf{R}^\tau := \mathbf{R}^\emptyset \cup \mathbf{R}^{\mathbf{T}\emptyset} \cup \mathbf{R}^{\mathbf{G}}$, the only new triples – terminological or not – that can arise in the computation of $\overline{\mathcal{G}}^\tau$ after deriving $\overline{\mathcal{T}}^\tau$ are from rules in $\mathbf{R}^{\mathbf{G}}$. □

We have shown that for an arbitrary ruleset and graph, the $\mathcal{T}$-split closure is sound wrt. the standard closure, and that if no T-Box triples are produced by rules requiring assertional knowledge, then $\mathcal{T}$-split closure is complete wrt. the standard closure. So, when are T-Box triples produced by $\mathbf{R}^{\mathbf{G}}$ rules? Analysis must be applied per ruleset, but for RDFS, pD* and OWL 2 RL/RDF, we informally posit that by inspection, one can show that such a condition can only arise through so called *non-standard usage* [8]: the assertion of terminological triples which use meta-classes and meta-properties in positions other than the object of `rdf:type` triples or predicate position respectively – e.g., `my:subPropertyOf rdfs:subPropertyOf rdfs:subPropertyOf .`

The $\mathcal{T}$-split approach can be implemented through partial indexing using two scans of the data: the first separates and builds the T-Box and the second reasons over the A-Box – note that the first scan can be over a separate T-Box graph. Algorithm 1 details this approach, which largely follows the formalisms in Definition 3: the major variance consists of the application of rules in $\mathcal{R}^{\mathbf{G}}$, which one can convince themselves is equivalent since *all* triples encountered are passed through every rule in $\mathcal{R}^{\mathbf{G}}$. For brevity, we omit some implementational details such as partial duplicate detection implemented using an LRU locality cache. The "non-trivial" aspects of the implementation include the indexing of the T-Box $\overline{\mathcal{T}}^\tau$, and the indexing of the A-Box $\mathcal{A}$. Again, as $\mathcal{A}$ is required to store more data, the two-scan approach becomes more inefficient than the "full-indexing" approach; in particular, a rule in $\mathcal{R}^{\mathbf{G^n}}$ with an open pattern – e.g., OWL 2 RL/RDF rule **eq-rep-s**: `(?s,o:sameAs,?s')` $\wedge$ `(?s,?p,?o)` $\Rightarrow$ `(?s',?p,?o)` – will require indexing of all data, negating the benefits of the approach. Again, partial-indexing performs well if $\mathcal{A}$ remains small and performs best if $\mathcal{R}^{\mathbf{G^n}} = \emptyset$ – i.e., no rules require A-Box joins and thus A-Box indexing is not required.

Algorithm 1: Partial indexing approach for $\mathcal{T}$-split closure

```
    Required: $\mathcal{R}, \mathcal{G}$
1   $\mathcal{R}^\tau := \tau(\mathcal{R}); \mathcal{T}_0 := \mathbf{T}(\mathcal{A}x, \mathcal{R}^\tau); n := 0;$            /* get t-split rules & ax. T-Box triples */
2   for $t \in \mathcal{G}$ do $\mathcal{T}_0 := \mathcal{T}_0 \cup \mathbf{T}(\{t\}, \mathcal{R}^\tau)$ ;            /* SCAN 1: extract T-Box from main data */
3   while $\mathcal{T}_{n+1} \neq \mathcal{T}_n$ do $\mathcal{T}_{n+1} := \mathcal{T}_n \cup \mathbf{T}(T_{\mathcal{R}\mathbf{T}\emptyset}(\mathcal{T}_i, \emptyset), \mathcal{R}^\tau); n$++ ;            /* do T-Box reasoning */
4   $\overline{\mathcal{T}}^\tau := \mathcal{T}_{n+1}; \overline{\mathcal{G}}^\tau := \mathcal{G}_0^\tau := \mathcal{G} \cup \overline{\mathcal{T}}^\tau \cup \mathcal{A}x; \mathcal{A} := \emptyset;$            /* initialise A-Box structures */
5   for $t^I \in \mathcal{G}_0^\tau$ do            /* SCAN 2: A-Box reasoning over all data */
6       $\mathcal{G}_0^I := \emptyset; \mathcal{G}_1^I := \{t^I\}; n := 1;$            /* initialise set to hold inferences from $t^I$ */
7       while $\mathcal{G}_n^I \neq \mathcal{G}_{n-1}^I$ do            /* while we find new triples to reason over */
8           for $t \in \mathcal{G}_n^I \setminus \mathcal{G}_{n-1}^I$ do            /* scan new triples */
9               $\mathcal{G}_{n+1}^I := \mathcal{G}_n^I \cup T_{\mathcal{R}\mathbf{G1}}(\overline{\mathcal{T}}^\tau, \{t\});$            /* do all ‘no A-Box join’ rules for $t$ */
10              for $r \in \mathcal{R}^{\mathbf{Gn}}$ do            /* for each ‘A-Box join’ rule */
11                  for $t^v \in \mathcal{A}nte_r^{\mathcal{G}}$ do            /* for each assertional pattern */
12                      if $\exists \mu \in \mathbf{M} : \mu(t^v) = t$ then $\mathcal{A} := \mathcal{A} \cup \{t\}$ ;            /* index $t$ if needed */
13                  $\mathcal{G}_{n+1}^I := \mathcal{G}_{n+1}^I \cup T_r(\overline{\mathcal{T}}^\tau, \mathcal{A});$            /* apply ‘A-Box join’ rule over index */
14          $n$++;            /* recurse */
15      $\overline{\mathcal{G}}^\tau := \overline{\mathcal{G}}^\tau \cup \mathcal{G}_n^I;$            /* write set of recursive inferences for $t^I$ to output */
    Return : $\overline{\mathcal{G}}^\tau$
```

4 Template Rules

We now discuss optimisations for deriving $\mathcal{T}$-split closure based on template rules, which are currently used by DLEJena [13] and also used in RIF for supporting OWL 2 RL/RDF [15]; however, instead of manually specifying a set of template rules, we leverage our general notion of terminological data to create a generic template function: after separating and closing the T-Box, we bind the T-Box patterns of rules before accessing the A-Box to create a set of new *templated rules* (or $\mathcal{T}$-ground rules) which themselves ‘encode’ the T-Box, thus avoiding repetitive T-Box pattern bindings during the A-Box reasoning process. We now formalise these notions.

Definition 4. Template Function *For a $\mathcal{T}$-split rule r^τ, the template function is given as* $\alpha : \mathbf{R}^\tau \times 2^{\mathbf{G}} \to 2^{\mathbf{R}}$; $(r^\tau, \mathcal{T}) \mapsto \{(\mu(\mathcal{A}nte_{r^\tau}^{\mathcal{G}}), \mu(\mathcal{C}on_{r^\tau})) \mid \mu \in \mathbf{M}(\mathcal{A}nte_{r^\tau}^{\mathcal{T}}, \mathcal{T})\}$.

Example 2. Given a simple T-Box $\mathcal{T} := \{($`f:Person`,`r:subClassOf`,`f:Agent`$)\}$ and a rule $r^\tau := \underline{(\texttt{?c1,r:subClassOf,?c2})} \wedge (\texttt{?x,a,?c1}) \Rightarrow (\texttt{?x,a,?c2})$, then the template function is given as $\alpha(r^\tau, \mathcal{T}) := \{(\texttt{?x,a,f:Person}) \Rightarrow (\texttt{?x,a,f:Agent})\}$.

Templated rule application is synonymous with standard rule application. We may use α as intuitive shorthand to map a *set* of $\mathcal{T}$-split rules to the union of the set of resulting templated rules. We now (i) propose that applying a $\mathcal{T}$-split rule gives the same result as applying the respective set of templated rules wrt. arbitrary graphs $\mathcal{T}$ & $\mathcal{G}$; (ii) describe the closure of a graph using templated rules; (iii) show that the templated-rule closure equals the $\mathcal{T}$-split closure previously outlined.

Proposition 3. *For any graphs $\mathcal{T}, \mathcal{G}$ and for any rule r with a $\mathcal{T}$-split rule $r^\tau = \tau(r)$, it holds that $T_{r^\tau}(\mathcal{T}, \mathcal{G}) = T_{\alpha(r^\tau, \mathcal{T})}(\mathcal{G})$.*

Proof. $T_{r^\tau}(\mathcal{T}, \mathcal{G}) = \bigcup_{\mu_0 \in \mathbf{M}(\mathcal{A}nte_{r^\tau}^{\mathcal{T}}, \mathcal{T})} \bigcup_{\mu_1 \in \mathbf{M}(\mu_0(\mathcal{A}nte_{r^\tau}^{\mathcal{G}}), \mathcal{G})} (\mu_0 \circ \mu_1)(\mathcal{C}on_{r^\tau}) = \bigcup_{r \in \alpha(r^\tau, \mathcal{T})} \bigcup_{\mu \in \mathbf{M}(\mathcal{A}nte_r, \mathcal{G})} \mu(\mathcal{C}on_r) = T_{\alpha(r^\tau, \mathcal{T})}(\mathcal{G})$. □

Definition 5. Templated rule closure *Given a ruleset $\mathcal{R}$, its $\mathcal{T}$-split version $\mathcal{R}^\tau := \tau(\mathcal{R})$, and a graph $\mathcal{G}$, let $\overline{\mathcal{T}}^\tau$ represent the closed T-Box as derived in the $\mathcal{T}$-split closure, and let $\mathcal{R}^\alpha := \alpha(\mathcal{R}^{\mathbf{G}}, \overline{\mathcal{T}}^\tau)$. Again, let $\mathcal{G}_0^\alpha := \mathcal{G} \cup \overline{\mathcal{T}}^\tau \cup \mathcal{A}x$, but this time $\mathcal{G}_{i+1}^\alpha := \mathcal{G}_i^\alpha \cup T_{\mathcal{R}^\alpha}(\mathcal{G}_i^\alpha)$; as before,* the templated rule closure *is $\mathcal{G}_n$ for the smallest value of n such that $\mathcal{G}_n^\alpha = T_{\mathcal{R}^\alpha}(\mathcal{G}_n^\alpha)$, denoted as $Cl_{\mathcal{R}^\alpha}(\mathcal{T}, \mathcal{G}^\alpha)$, or simply $\overline{\mathcal{G}}^\alpha$.*

Theorem 3. *For any graph $\mathcal{G}$, and any ruleset $\mathcal{R} \subset \mathbf{R}$, its $\mathcal{T}$-split version $\mathcal{R}^\tau$, and the respective templated ruleset $\mathcal{R}^\alpha$, we can say that $\overline{\mathcal{G}}^\alpha = \overline{\mathcal{G}}^\tau$.*

Proof. The only divergence between the $\mathcal{T}$-split closure and templated-rule closure is in the fixpoint calculation: $\mathcal{G}_{i+1}^\alpha := \mathcal{G}_i^\alpha \cup T_{\mathcal{R}^\alpha}(\mathcal{G}_i^\alpha)$ versus $\mathcal{G}_{i+1}^\tau := \mathcal{G}_i^\tau \cup T_{\mathcal{R}^{\mathbf{G}}}(\overline{\mathcal{T}}^\tau, \mathcal{G}_i^\tau)$. Using induction, by def. $\mathcal{G}_0^\tau = \mathcal{G}_0^\alpha$; if $\mathcal{G}_i^\tau = \mathcal{G}_i^\alpha$, then $\mathcal{G}_{i+1}^\tau = \mathcal{G}_i^\alpha \cup \bigcup_{r^\tau \in \mathcal{R}^{\mathbf{G}}} T_{r^\tau}(\overline{\mathcal{T}}^\tau, \mathcal{G}_i^\alpha) = \mathcal{G}_i^\alpha \cup \bigcup_{r \in \alpha(\mathcal{R}^{\mathbf{G}}, \overline{\mathcal{T}}^\tau)} T_{r^\tau}(\mathcal{G}_i^\alpha) = \mathcal{G}_i^\alpha \cup T_{\mathcal{R}^\alpha}(\mathcal{G}_i^\alpha) = \mathcal{G}_{i+1}^\alpha$. □

The templated rules can be applied in lieu of the $\mathcal{R}^{\mathbf{G}}$ rules in Algorithm 1. Indeed, a large number of rules can be templated for a sufficiently complex T-Box, and naïve application of all such rules on all triples could worsen performance; however, the templated rules are more amenable to further optimisations, which we now discuss.

4.1 Merging Equivalent Template Rules

The templating procedure may result in rules with equivalent antecedents – which can be aligned by variable rewriting – being produced; these rules can subsequently be merged. We formalise such notions here.

Definition 6. Equivalent Graph Patterns: *Let $\mathbf{N}$ be the set of automorphic* variable rewrite mappings *containing all ν as follows:*

$$\nu : \mathbf{V} \cup \mathbf{C} \rightarrowtail\!\!\!\!\twoheadrightarrow \mathbf{V} \cup \mathbf{C};\ x \mapsto \begin{cases} x & \textit{if } x \in \mathbf{C} \\ v \in \mathbf{V} & \textit{otherwise} \end{cases} \tag{2}$$

(Note: $\mathbf{N} \subset \mathbf{M}$). We denote by $\sim_\nu$ an equivalence relation for graph patterns *such that $\mathcal{G}_i^{\mathbf{V}} \sim_\nu \mathcal{G}_j^{\mathbf{V}}$ iff there exists a mapping $\nu \in \mathbf{N}$ such that $\nu(\mathcal{G}_i^{\mathbf{V}}) = \mathcal{G}_j^{\mathbf{V}}$.*

Proposition 4. *The relation $\sim_\nu$ is reflexive, symmetric and transitive.*

Proof. Reflexivity is trivially given by the identity morphism $\nu(x) = x$, symmetry is given by the inverse morphism $\nu^{-1}(\mathcal{G}_j^{\mathbf{V}})$ where $\nu^{-1} \in \mathbf{N}$ if $\nu \in \mathbf{N}$ since ν is automorphic, and transitivity is given by the presence of the composite morphism $(\nu_a \circ \nu_b)(\mathcal{G}^{\mathbf{V}})$ where again $\nu_a \circ \nu_b \in \mathbf{N}$ since ν_a and ν_b are automorphic. □

Definition 7. Rule Merge: *Let $\sim_{\mathbf{R}}$ be an equivalence relation – slightly abusing notation – which holds between two rules such that $r_i \sim_{\mathbf{R}} r_j$ iff $\mathcal{A}nte_{r_i} \sim_\nu \mathcal{A}nte_{r_j}$. Given an equivalence class $[r]$ – a set of rules between which $\sim_{\mathbf{R}}$ holds – select a* canonical rule *$r \in [r]$; we can now describe the merge of the equivalence class as $\beta([r]) := (\mathcal{A}nte_r, \mathcal{C}on_{[r]})$ where $\mathcal{C}on_{[r]} := \bigcup_{r_i \in [r]} \nu_i(\mathcal{C}on_{r_i})$ for some $\nu_i \in \mathbf{N}$ such that $\nu_i(\mathcal{A}nte_{r_i}) = \mathcal{A}nte_r$. Now letting $\mathcal{R}/{\sim_{\mathbf{R}}} := \{[r] \mid r \in \mathcal{R}\}$ denote the quotient set of $\mathcal{R}$ by $\sim_{\mathbf{R}}$ – the set of all equivalent classes $[r]$ wrt. $\sim_{\mathbf{R}}$ in $\mathcal{R}$ – we can generalise the* rule merge function *for a set of rules as $\beta : 2^{\mathbf{R}} \to 2^{\mathbf{R}}, \mathcal{R} \mapsto \bigcup\{\beta([r]) \mid [r] \in \mathcal{R}/{\sim_{\mathbf{R}}}\}$.*

Example 3. Taking the two templated rules: (`?x,f:img,?y`) $\Rightarrow$ (`?x,a,f:Person`) and (`?s,f:img,?o`) $\Rightarrow$ (`?s,f:depicts,?o`); they can be merged by ν where ν(`?s`) = `?x`, ν(`?o`) = `?y`, giving (`?x,f:img,?y`) $\Rightarrow$ (`?x,a,f:Person`) $\wedge$ (`?x,f:depicts,?y`).

The choice of canonical rule is unimportant since ν is automorphic; we now show that the application of any ruleset and the respective merged ruleset are extensionally equal.

Proposition 5. *For any graph $\mathcal{G}$ and $\sim_{\mathbf{R}}$ equivalence class $[r]$, $T_{[r]}(\mathcal{G}) = T_{\beta([r])}(\mathcal{G})$; for any ruleset $\mathcal{R}$, $T_{\mathcal{R}}(\mathcal{G}) = T_{\beta(\mathcal{R})}(\mathcal{G})$; wrt. closure, $Cl_{\mathcal{R}^{\alpha}}(\mathcal{T},\mathcal{G}) = Cl_{\beta(\mathcal{R}^{\alpha})}(\mathcal{T},\mathcal{G})$.*

Proof. We denote $\beta([r])$ as $(\mathcal{A}nte_{\beta}, \mathcal{C}on_{\beta})$. If $\mathcal{G}_i^{\mathbf{V}} \sim_{\nu} \mathcal{G}_j^{\mathbf{V}}$, then by def. $\nu(\mathcal{G}_i^{\mathbf{V}}) = \mathcal{G}_j^{\mathbf{V}}$, and for any graph $\mathcal{G}$ and any mapping $\mu \in \mathbf{M}$, $\mu(\nu(\mathcal{G}_i^{\mathbf{V}})) = \mu(\mathcal{G}_j^{\mathbf{V}})$; i.e., if $\mathcal{G}_i^{\mathbf{V}} \sim_{\nu} \mathcal{G}_j^{\mathbf{V}}$, $\mathbf{M}(\nu(\mathcal{G}_i^{\mathbf{V}}),\mathcal{G}) = \mathbf{M}(\mathcal{G}_j^{\mathbf{V}},\mathcal{G})$. Thus we give $\mathcal{M}_{\beta} := \{\mu \mid \mu(\mathcal{A}nte_{\beta}) \subseteq \mathcal{G}\} = \bigcup_{r_i \in [r]}\{\mu \mid \mu(\nu_i(\mathcal{A}nte_{r_i})) \subseteq \mathcal{G}\}$. Let $\mathcal{M}_i := \{\mu \mid \mu(\mathcal{A}nte_{r_i}) \subseteq \mathcal{G}\}$; now, it follows that $T_{\beta([r])}(\mathcal{G}) = \bigcup_{\mu \in \mathcal{M}_{\beta}} \mu(\mathcal{C}on_{\beta}) = \bigcup_{r_i \in [r]} \bigcup_{\mu \in \mathcal{M}_{\beta}} \mu(\nu_i(\mathcal{C}on_{r_i})) = \bigcup_{r_i \in [r]} \bigcup_{\mu \in \mathcal{M}_i} \mu(\mathcal{C}on_{r_i}) = T_{[r]}(\mathcal{G})$. The rest of the proposition follows naturally. $\square$

4.2 Rule Index

We have reduced the amount of templated rules through merging; however, given a sufficiently complex T-Box, we may still have a prohibitive number of rules for efficient recursive application. We now look at the use of a rule index which maps a triple t to rules containing an antecedent pattern which t is a binding for, thus enabling the efficient identification and application of only relevant rules for a given triple.

Definition 8. Rule Lookup *Given a triple t and ruleset $\mathcal{R}$, the* rule lookup function *is* $\omega : \mathbf{G} \times 2^{\mathbf{R}} \to 2^{\mathbf{R}}$, $(t,\mathcal{R}) \mapsto \{r \in \mathcal{R} \mid \exists \mu \in \mathbf{M} : \exists t^v \in \mathcal{A}nte_r : (\mu(t^v) = t)\}$.

Example 4. Given a triple $t :=$ (`ex:me`,`a`,`f:Person`), and a simple example ruleset $\mathcal{R} := \{$(`?x`,`f:img`,`?y`) $\Rightarrow$ (`?x`,`a`,`f:Person`); (`?x`,`a`,`f:Person`) $\Rightarrow$ (`?x`,`a`,`f:Agent`); (`?x`,`a`,`?y`) $\Rightarrow$ (`?y`,`a`,`r:Class`)$\}$, $\omega(t,\mathcal{R})$ returns a set containing the latter two rules.

A triple pattern has $2^3 = 8$ possible forms: $(?,?,?), (s,?,?), (?,p,?), (?,?,o), (s,p,?), (?,p,o), (s,?,o), (s,p,o)$. Thus, we require eight indices for antecedent triple patterns, and eight lookups to perform $\omega(t,\mathcal{R})$ – to find all relevant rules for a triple. We use seven in-memory hashtables storing the constants of the rule antecedent patterns as key, and a set of rules containing such a pattern as value; e.g., $\{$(`?x`,`a`,`f:Person`)$\}$ is put into the $(?,p,o)$ index with $\{$`a`,`f:Person`$\}$ as key. Rules containing patterns without constants are stored in a set, as they are relevant to all triples.

4.3 Rule Dependency – Labelled Rule Graph

Within our rule index, there may exist rule dependencies: the application of one rule may/will lead to the application of another. Thus, instead of performing lookups for rules for each recursively inferred triple, we can model dependencies in our rule index using a rule graph. In Logic Programming, a rule graph is defined as a directed graph $\mathcal{H} := (\mathcal{R}, \Omega)$ where $(r_i, r_j) \in \Omega$ (i.e., $r_i \,\Omega\, r_j$, read "r_j follows r_i") iff there exists a mapping $\mu \in \mathbf{M}$ such that $\mu(t^v) \in \mathcal{C}on_{r_i}$ for $t^v \in \mathcal{A}nte_{r_j}$ (cf. [14]).

By building and encoding such a rule graph into our index, we can "wire" the recursive application of rules for a given triple. However, from the merge function (or otherwise) there may exist rules with large consequent sets. We therefore extend the notion of the rule graph to a directed labelled graph with inclusion of the labelling function $\lambda : \mathbf{R} \times \mathbf{R} \rightarrow 2^{\mathbf{G^V}}; (r_i, r_j) \mapsto \{t^v \in \mathcal{C}on_{r_i} \mid \exists \mu \in \mathbf{M} : \mu^{-1}(t^v) \in \mathcal{A}nte_{r_j}\}$; in simpler terms, $\lambda(r_i, r_j)$ gives the set of consequent triple patterns in r_i that would be matched by patterns in the antecedent of r_j, labelling the edges Ω of the rule graph with the consequent patterns that give the dependency.

Example 5. For a rule $r_i :=$ (`?x,f:img,?y`) $\Rightarrow$ (`?x,a,f:Person`) $\wedge$ (`?y,a,f:Image`), and a rule $r_j :=$ (`?s,a,f:Person`) $\Rightarrow$ (`?s,a,f:Agent`), we say that $r_i \, \Omega \, r_j$, where $\lambda(r_i, r_j) = \{$(`?x,a,f:Person`)$\}$.

In practice, our rule index stores sets of elements of a linked list, where each element contains a rule and links to rules which are relevant for that rule's consequent patterns. Thus, for each input triple, we can retrieve all relevant rules for all eight possible patterns, apply those rules, and if successful, follow the respective labelled links to recursively find relevant rules without re-accessing the index until the next input triple.

4.4 Rule Saturisation

We very briefly describe one final and intuitive optimisation technique we investigated – which later evaluation demonstrates to be mostly disadvantageous – involving the *saturisation of rules*; we say that a subset of dependencies in the rule graph are *strong dependencies*, where the successful application of one rule *will* always lead to the successful application of another. For linear rules, we can saturate rules by pre-computing the recursive rule application of its dependencies; we give the gist with an example:

Example 6. Take rules $r_i :=$ (`?x,f:img,?y`) $\Rightarrow$ (`?x,a,f:Person`) $\wedge$ (`?y,a,f:Image`), $r_j :=$ (`?s,a,f:Person`) $\Rightarrow$ (`?s,a,f:Agent`), $r_k :=$ (`?x,a,?y`) $\Rightarrow$ (`?y,a,r:Class`)$\}$. We can see that $r_i \, \Omega \, r_j$, $r_i \, \Omega \, r_k$, $r_j \, \Omega \, r_k$. We can remove the links from r_i to r_j and r_k (and similarly from r_j to r_k) by saturating r_i to (`?x,f:img,?y`) $\Rightarrow$ (`?x,a,f:Person`) $\wedge$ (`?y,a,f:Image`)$\wedge$(`?x,a,f:Agent`)$\wedge$(`f:Person,a,r:Class`)$\wedge$(`f:Image,a,r:Class`)$\wedge$ (`f:Agent,a,r:Class`)$\}$.

As we will see in Sections 4.6 & 5.2, saturisation produces more duplicates and thus puts more load on the duplicate-removal cache, negatively affecting performance.

4.5 Optimised Partial Indexing Approach using Template Rules

We now integrate the above notions as optimisations for the partial indexing approach, with the new procedure detailed in Algorithm 2. We no longer need to bind T-Box patterns during A-Box access; we mitigate the cost of extra templated rules by first merging rules, and instead of brute-force applying all rules to all triples in the A-Box reasoning scan, we use our linked rule index to retrieve only relevant rules for a given triple and to find recursively relevant rules. We now initially evaluate our methods.

4.6 Preliminary Performance Evaluation

In order to initially evaluate the above optimisations, we applied small-scale reasoning for RDFS (minus the infinite `rdf:_n` axiomatic triples [4]), pD* and OWL 2 RL/RDF over LUBM(10) [3], consisting of about 1.3m triples – note that we do exclude **lg/gl** rules for RDFS/pD* since we allow generalised triples [2]. All evaluation in this paper has been run on single-core 2.2GHz Opteron x86-64 machine(s) with 4GB of main memory. Table 1 gives the performance for the following partial-indexing configurations: (i) N: 'normal' $\mathcal{T}$-split closure; (ii) NI: normal $\mathcal{T}$-split closure with linked rule

Algorithm 2: Partial-indexing approach using templated rule optimisations

```
   Required: $\mathcal{R}, \mathcal{G}$
1  derive $\overline{\mathcal{T}}^\tau$ and $\mathcal{R}^\tau$ as in Algorithm 1;                          /* SCAN 1: See Algorithm 1 */
2  $\mathcal{R}^\alpha := \alpha(\mathcal{R}^{\mathbf{G}}); \mathcal{R}^\beta := \beta(\mathcal{R}^\alpha)$;                       /* template and merge $\mathcal{T}$-split rules */
3  build $\omega$ index for $\mathcal{R}^\beta$ encoding graph $\mathcal{H}$ with edges $\lambda$;        /* build rule index w/ dependencies */
4  $\overline{\mathcal{G}}^\tau := \mathcal{G}^\tau_0 := \mathcal{G} \cup \overline{\mathcal{T}}^\tau \cup \mathcal{A}x; \mathcal{A} := \emptyset$;                        /* init A-Box structures */
5  for $t^I \in \mathcal{G}^\tau_0$ do                                   /* SCAN 2: A-Box reasoning over all data */
6      $\mathcal{RG}^I_0 := \emptyset; \mathcal{RG}^I_1 := \{(r, t^I) \mid r \in \omega(t^I, \mathcal{R}^\beta)\}; n := 1$;      /* initialise relevant rules for $t^I$ */
7      while $\mathcal{RG}^I_n \neq \mathcal{RG}^I_{n-1}$ do          /* while we find new rule/triple pairs to reason over */
8          for $(r, t) \in \mathcal{RG}^I_n \setminus \mathcal{RG}^I_{n-1}$ do                      /* scan new rule/triple pairs */
9              $\mathcal{G}_{rt} := \emptyset; \mathcal{RG}^I_{n+1} := \mathcal{RG}^I_n$;                 /* initialise state for rule triple pair */
10             if $|\mathcal{A}nte_r| > 1$ then                                  /* if rule requires A-Box join */
11                 for $t^v \in \mathcal{A}nte^{\mathcal{G}}_r$ do                          /* for each assertional pattern */
12                     if $\exists \mu \in \mathbf{M} : \mu(t^v) = t$ then $\mathcal{A} := \mathcal{A} \cup \{t\}$ ;              /* index $t$ if needed */
13                 $\mathcal{G}_{rt} := T_r(\overline{\mathcal{T}}^\tau, \mathcal{A})$;                       /* apply 'A-Box join rule' over index */
14             else
15                 $\mathcal{G}_{rt} := T_r(\overline{\mathcal{T}}^\tau, \{t\})$;                    /* apply 'non A-Box join rule' for $t$ */
16             if $\mathcal{G}_{rt} \neq \emptyset$ then                                /* if rule creates inference */
17                 for $r^+ : (r, r^+) \in \Omega$ do                         /* find successive rules in graph */
18                     for $t^v_n \in \lambda(r, r^+)$ do                 /* for the consequent patterns bound */
19                         $\mathcal{RG}^I_{n+1} := \mathcal{RG}^I_{n+1} \cup \{(r^+, t_n) \mid t_n \in \mathcal{G}_{rt}\}$;   /* add rule/triple pair */
20         $n$++;                                                  /* recurse for unique rule/triple pair */
21     $\overline{\mathcal{G}}^\tau := \overline{\mathcal{G}}^\tau \cup \{t \mid (r, t) \in \mathcal{RG}^I_n\}$;                /* write recursive inferences for $t^I$ to output */
   Return : $\overline{\mathcal{G}}^\tau$
```

`input`	**LUBM(10) - 1.27M data triples, 295 ontology triples**																	
`fragment`	**RDFS**						**pD***						**OWL 2 RL**					
`inferred`	748k						1,328k						1,597k					
`tmpl. rules`	149						175						378					
`after merge`	87						108						119					
`config.`	*N*	*NI*	*T*	*TI*	*TIM*	*TIMS*	*N*	*NI*	*T*	*TI*	*TIM*	*TIMS*	*N*	*NI*	*T*	*TI*	*TIM*	*TIMS*
`time (s)`	99	117	404	89	81	69	365	391	734	227	221	225	858	940	1,690	474	443	465
`rule apps (m)`	16.5	15.5	308	11.3	9.9	7.8	62.5	50	468	22.9	21.1	13.9	149	110	1,115	81.8	78.6	75.6
`% success`	43.4	46.5	2.4	64.2	62.6	52.3	18.8	23.4	2.6	51.5	48.7	61.3	4.2	5.6	0.8	10.5	6.8	15
`cache hit (m)`	10.8	10.8	8.2	8.2	8.2	8.1	19.1	19.1	15.1	15.1	14.9	38.7	16.5	16.5	13.1	13	12.7	34.4

Table 1. Details of reasoning for LUBM(10) given different reasoning configurations.

index; (iii) T: $\mathcal{T}$-split closure wrt. templated rules; (iv) TI: $\mathcal{T}$-split closure wrt. linked templated rule index; (v) TIM: $\mathcal{T}$-split closure wrt. linked & merged templated rule index; (vi) TIMS: $\mathcal{T}$-split closure wrt. linked, merged & saturated templated rule index.

In all approaches, exhaustively applying templated rules demonstrates the worst performance; after indexing the approach becomes the most efficient. RDFS gains little in the way of improvement, but in fact only contains 8 rules requiring A-Box data: the reduction in rule applications given by templating and indexing is modest. OWL 2 RL and pD* take just over half the time for TI* vs. N* approaches. A correlation between increased rule applications and increased inferencing time is evident, but sometimes fails: e.g., for pD*, TIMS gives less rule applications than TIM, but takes more time – in such cases, we see the cache encountering more duplicates – as mentioned, saturated rules can immediately produce a batch of duplicates that would otherwise halt a chain of inferences mid-way. OWL 2 RL creates more templated rules than pD* due to expanded T-Box level reasoning, but these are merged to a number just above pD*: OWL 2 RL supports intersection-of inferencing used by LUBM and not in pD*. LUBM does not contain OWL 2 constructs, but redundant rules are factored out during templating.

Although we improve the performance of pD* and OWL 2 RL/RDF inferencing, we perform A-Box joins in-memory, and in fact cannot scale much beyond the limited scale above for these fragments: again our optimisations focus on linear rules. We now reunite with our original use-case of Linked Data reasoning, focussing on the application of linear rules and shifting up three orders of magnitude.

5 Reasoning for Linked Data

Again, we aim at reasoning over Linked Data for the SWSE project. In previous works, we have investigated the unique challenges of reasoning over the open Web, and identified the need for scale, incompleteness, and consideration of the source of data. In [8], we applied reasoning over 1 billion Linked Data triples *using T-Box optimisations specific to a subset of pD**; we (i) demonstrated that aside from equality reasoning, pD* rules which do not require A-Box joins covered 99% of inferences possible in our Web dataset, based on the observation that the most commonly instantiated vocabularies on the Web typically use lightweight RDFS and OWL terms supportable by linear rules; (ii) discussed the dangers of applying materialisation over open Web data, which can naïvely lead to an explosion of inferences: for example, one document[3] defines `owl:Thing` to be a member of 55 union classes, another defines nine *properties* as the domain of `rdf:type`[4], etc. Observation (i) ties in with our linear-rule optimisations; however, equality reasoning requires A-Box joins: we see `owl:sameAs` related inferencing as very important for data integration within the Linked Data use-case, but prefer a decoupling of such reasoning – which entails its own requirements and challenges – and have analysed the issue separately in previous works [10]. Observation (ii) motivates our next discussion: we now reintroduce our notion of *authoritative reasoning*.

5.1 Authoritative Reasoning

In order to curtail the possible side-effects of open Web data publishing, we include the source of data in inferencing. Our methods are based on the view that a publisher instantiating a vocabulary's term (class/property) thereby accepts the inferencing mandated by that vocabulary (and recursively referenced vocabularies) for that term. Thus, once a publisher instantiates a term from a vocabulary, only that vocabulary and its references should influence what inferences are possible through that instantiation.

Firstly, we must define the relationship between a term and a vocabulary. We view a term as an RDF constant, and a vocabulary as a Web document: we give the function $\mathit{http} : \mathbf{U} \rightarrow 2^{\mathbf{G}}$ as the mapping from a URI (a Web location) to an RDF graph it may provide by means of a given HTTP lookup. In Linked Data principles, dereferencable URIs are encouraged; dereferencing can be seen as a function $deref : \mathbf{U} \rightarrow \mathbf{U}$ which maps one URI to another by means of HTTP dereferencing mechanisms (this may include removal of a URI fragment identifier and recursive but finite redirects, and maps a URI to itself in case of failure; such functions are fixed to the time the data was crawled).

[3] http://lsdis.cs.uga.edu/ oldham/ontology/wsag/wsag.owl
[4] http://www.eiao.net/rdf/1.0

We then give the authoritative function:

$$auth : \mathbf{U} \to 2^{\mathbf{C}};\ u \mapsto \{c \mid c \in \mathbf{B}, c \in t \in \mathit{http}(u) \text{ or } c \in \mathbf{U}, \mathit{deref}(c) = u\} \qquad (3)$$

where a Web document is authoritative for URIs which dereference to it and the blank nodes it contains; e.g., the FOAF vocabulary is authoritative for terms in its namespace.

To negate the effects of non-authoritative axioms on reasoning over Web data, we apply restrictions to the $\mathcal{T}$-split rule application of rules in $\mathbf{R^{TG}}$, whereby, for the mapping μ of the rule application as before, there must additionally exist a $\mu(v)$ such that $v \in \mathbf{V}(\mathcal{A}nte^{\mathcal{T}}) \cap \mathbf{V}(\mathcal{A}nte^{\mathcal{G}}), \mu(v) \in auth(u), \mu(\mathcal{A}nte^{\mathcal{T}}) \subseteq \mathit{http}(u)$.[5]

Example 7. Take rule r^{τ} := $\underline{\texttt{(?c1,r:subClassOf,?c2)}}\wedge$(?x,a,?c1) $\Rightarrow$ (?x,a,?c2). Here, $\mathbf{V}(\mathcal{A}nte^{\mathcal{T}}_{r^{\tau}}) \cap \mathbf{V}(\mathcal{A}nte^{\mathcal{G}}_{r^{\tau}}) = \{$?c1$\}$. Take an A-Box triple (ex:me,a,f:Person); μ(?c1) = f:Person. Let deref(f:Person) = f: the FOAF spec; now, $\{u \mid \mu($?c1$) \in auth(u)\} = \{$f:$\}$. Any triple of the form (f:Person,r:subClassOf,?c2) must come from f: for the rule to be authoritatively applied. Note that ?c2 can be arbitrarily bound; i.e., FOAF can *extend* any classes they like.

We refer the reader to [8] for more detail on authoritative reasoning. Note that the previous two examples from documents in Footnotes 3 & 4 are ignored by the authoritative reasoning. Since authoritativeness is on a T-Box level, we can apply the above additional restriction to our templating function when binding the terminological patterns of the rules to derive a set of *authoritative templated rules*.

5.2 Linked Data Reasoning Evaluation

We now give evaluation over 1.12b quads (947m unique triples) of Linked Data crawled for SWSE in May 2010. Note that we use a GZip compressed file of quadruples as input to the reasoning process: the fourth element element encodes the provenance (Web source) of the contained triple; we also require information about redirects encountered in the crawl to reconstruct the deref function. We output a flat file of GZipped triples. We perform reasoning over a subset of OWL 2 RL/RDF containing 42 rules: firstly, we omit datatype reasoning which can lead to the inference of near-infinite triples (e.g., 1.000^^xsd:float owl:sameAs 1.00^^xsd:float); secondly, we currently omit inconsistency checking rules (we will examine use-cases for these rules in later work); thirdly, we omit rules which infer 'tautologies' – statements that hold for every term in the graph, such as reflexive owl:sameAs statements (we also filter these from the output). Given our use-case SWSE, we wish to infer a circumspect amount of data with utility for query-answering – completeness is not a requirement (cf. [5] for related discussion). For reasons of efficiency as described, we omit rules which require A-Box joins. Thus, our subset consists of the OWL 2 RL/RDF axiomatic rules, 'schema rules'[2, Table 9], and rules with one assertional pattern which we give in Table 3.

Reasoning over the dataset described inferred 1.58b raw triples, which were filtered to 1.14b triples removing non-RDF generalised triples and 'tautological statements' – post-processing revealed that 962m ($\sim$61%) were unique and had not been asserted

[5] Note here that we restrict the T-Box segment of a $\mathbf{R^{TG}}$ rule to be instantiated by one document; this is not so restrictive where in OWL 2 RL/RDF, all such rules contain one 'T-Box axiom', possibly described using multiple triples; cf. [8]. Also, we do not consider the results of T-Box level reasoning as authoritative.

	T-Box (min)	*A-Box (hr)*
N	8.9	118.4
NI	8.9	121.3
T	8.9	171609[a]
TI	8.9	22.1
TIM	8.9	17.7
TIMS	8.9	19.5

[a] Estimated as a linear product from one day of reasoning.

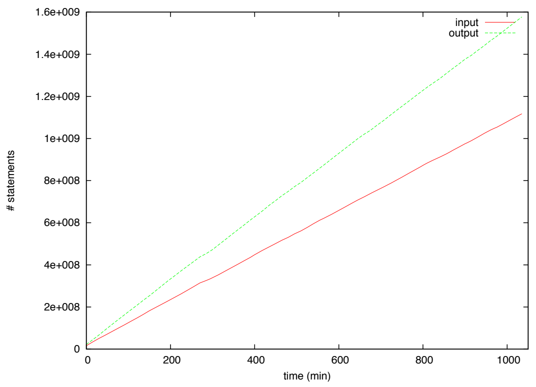

Fig. 1. Performance for reasoning over 1.1B statements on one machine for all approaches *(left)*, and detailed throughput performance for A-Box reasoning using the fastest approach *TIM (right)*.

Machines	*Extract T-Box*	*Build T-Box*	*Reason A-Box*	*Total*
1	492	8.9	1062	1565
2	240	10.2	465	719
4	131	10.4	239	383
8	67	9.8	121	201

Table 2. Distributed reasoning in minutes using TIM for 1, 2, 4 & 8 machines.

(roughly a 1:1 *reasoned*:*asserted* ratio). The first step – extracting 1.1m T-Box triples from the dataset – took 8.2 hrs. Subsequently, Figure 1 gives the results for reasoning on one machine for each approach as before. T-Box level processing – e.g., templating, rule indexing, etc. – took roughly the same time. For A-Box reasoning, saturation causes the same problems with extra duplicate triples as before, and so the fastest approach is TIM, which takes ∼15% of the time for the naïve $\mathcal{T}$-split closure algorithm; we also show the linear performance of TIM in Figure 1 (we would expect all methods to be similarly linear). 301k templated rules with 2.23m links are merged to 216k with 1.15m links; after saturation, each rule has an average of 6 consequent patterns and all links are successfully removed. Note that with 301k templated rules without indexing, applying all rules to all statements would take approx. 19 years.

Since all of our rules are linear, we can also distribute our approach by flooding the templated rules to all machines. In Table 2, we give the performance of such an approach for 1, 2, 4, and 8 machines using a simple RMI architecture [9]. Note that the most expensive aspects of the reasoning process – extracting the T-Box from the dataset and reasoning over the A-Box – can be executed independently in parallel. The only communication required between machines is the aggregation of the T-Box, and creation of the shared templated-rule index: this takes ∼10 mins, and becomes the lower bound for time taken for distributed evaluation with arbitrary machine count. In summary, we perform reasoning over 1.12b Linked Data triples in 3.35 hours using 8 machines, deriving 1.58b inferred triples, of which 962m are novel and unique.

6 Related Work

We have discussed our previous work on SAOR throughout the paper. Following initial work on SAOR – which had not yet demonstrated distribution – a number of scalable distributed reasoners adopted a similar approach to partial indexing herein reformalised. Weaver et al. [19] discuss a similar approach for distributed reasoning over RDFS; however, their experiments were solely over LUBM and their discussion was specific to RDFS. Urbani et al. [18] use MapReduce for distributed reasoning for RDFS over 850m Linked Data triples; they do not consider authority and produce 30b triples which is too much for our SWSE use-case – interestingly, they also tried pD* on 35m Web triples and stopped after inferring 3.8b inferences in 12 hours, lending strength to our arguments for authoritative reasoning. In very recent work, the same authors [17] apply incomplete but comprehensive pD* to 100b LUBM triples, discussing rule-specific optimisations for performing join rules over pD*: however, we feel that materialisation wrt. rules over 1b triples of arbitrary Linked Data is still an open research goal.

A viable alternative approach to Web reasoning employed by Sindice [1] – the relation to which is discussed in depth in [8] – is to consider merging small "per-document" closures which quarantines reasoning to a given document and the related documents it either implicitly or explicitly imports. Works on LDSR select clean subsets of Linked Data ∼0.9b triples and apply reasoning using the proprietary BigOWLIM reasoner [11].

With respect to template rules, DLEJena [13] uses the Pellet DL reasoner for T-Box level reasoning, and uses the results to template rules for the Jena rule engine; they only demonstrate methods on synthetic datasets up to a scale of ∼1M triples. We take a somewhat different direction, discussing optimisations for partial-indexing.

7 Conclusion

We have introduced the notion of terminological data for RDF(S)/OWL, and have generalised and formalised the notion of partial indexing techniques which are optimised for application of linear rules and which rely on a separation of terminological data – a non-recursive segment of the data; we then related the derived closure to semi-naïve evaluation. We subsequently discussed inclusion of a template function in such an algorithm, showing that naïvely, templated rules worsen performance, but with rule merging, indexing and linking techniques, templated rules outperform the base-line $\mathcal{T}$-split closure esp. for a complex T-Box. This work, along with DLEJena, supports uncited claims within the recently standardised RIF working group that rule templating offers a more efficient solution for supporting OWL 2 RL than a direct translation of OWL 2 RL/RDF rules [15, Section 1]. We then reintroduced some discussion relating to reasoning over Linked Data, including our notion of authoritativeness, and demonstrated scalable distributed reasoning over a subset of OWL 2 RL for 1.1b quads (without need for manual T-Box massaging or pre-selection). The SAOR system is actively used to provide reasoned data to the SWSE system [9] for live search and browsing over Linked Data: `http://swse.deri.org/`.

References

1. R. Delbru, A. Polleres, G. Tummarello, and S. Decker. Context Dependent Reasoning for Semantic Documents in Sindice. In *Proc. of 4th SSWS Workshop*, Oct. 2008.
2. B. C. Grau, B. Motik, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language: Profiles. W3C Recommendation, Oct. 2009.

$\mathrm{R}^{\mathrm{G}^1}$ **: only one assertional pattern in antecedent**			
OWL2RL	**Antecedent**		**Consequent**
	terminological	*assertional*	
eq-sym	-	?x owl:sameAs ?y .	?y owl:sameAs ?x .
prp-dom	**?p** rdfs:domain ?c .	?x ?p ?y .	?x a ?c .
prp-rng	**?p** rdfs:range ?c .	?x ?p ?y .	?y a ?c .
prp-symp	**?p** a owl:SymmetricProperty .	?x ?p ?y .	?y ?p ?x .
prp-spo1	**?p$_1$** rdfs:subPropertyOf ?p$_2$.	?x ?p$_1$?y .	?x ?p$_2$?y .
prp-eqp1	**?p$_1$** owl:equivalentProperty ?p$_2$.	?x ?p$_1$?y .	?x ?p$_2$?y .
prp-eqp2	?p$_1$ owl:equivalentProperty **?p$_2$** .	?x ?p$_2$?y .	?x ?p$_1$?y .
prp-inv1	**?p$_1$** owl:inverseOf ?p$_2$.	?x ?p$_1$?y .	?y ?p$_2$?x .
prp-inv2	?p$_1$ owl:inverseOf **?p$_2$** .	?x ?p$_2$?y .	?y ?p$_1$?x .
cls-int2	**?c** owl:intersectionOf (?c$_1$ ... ?c$_n$) .	?x a ?c .	?x a ?c$_1$...?c$_n$.
cls-uni	?c owl:unionOf (?c$_1$ ... **?c$_i$** ... ?c$_n$) .	?x a ?c$_i$	?x a ?c .
cls-svf2	?x owl:someValuesFrom owl:Thing ; owl:onProperty **?p** .	?u ?p ?v .	?u a ?x .
cls-hv1	**?x** owl:hasValue ?y ; owl:onProperty ?p .	?u a ?x .	?u ?p ?y .
cls-hv2	?x owl:hasValue **?y** ; owl:onProperty **?p** .	?u ?p ?y .	?u a ?x .
cax-sco	**?c$_1$** rdfs:subClassOf ?c$_2$.	?x a ?c$_1$.	?x a ?c$_2$.
cax-eqc1	**?c$_1$** owl:equivalentClass ?c$_2$.	?x a ?c$_1$.	?x a ?c$_2$.
cax-eqc2	?c$_1$ owl:equivalentClass **?c$_2$** .	?x a ?c$_2$.	?x a ?c$_1$.

Table 3. OWL 2 RL/RDF rules we apply *for Web reasoning* with exactly one assertional pattern. Authoritative variable positions are given in bold. Not shown are axiomatic and schema rules [2].

3. Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.
4. P. Hayes. RDF semantics. W3C Recommendation, Feb. 2004.
5. P. Hitzler and F. van Harmelen. A Reasonable Semantic Web. *Semantic Web Journal*, 1(1), 2010. (to appear – available from http://www.semantic-web-journal.net/).
6. A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. Weaving the Pedantic Web. In *Proc. of 3rd LDOW Workshop*, Apr. 2010.
7. A. Hogan, A. Harth, and A. Polleres. SAOR: Authoritative Reasoning for the Web. In *Proc. of 3rd ASWC*, Dec. 2008.
8. A. Hogan, A. Harth, and A. Polleres. Scalable Authoritative OWL Reasoning for the Web. *Int. J. Semantic Web Inf. Syst.*, 5(2), 2009.
9. A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker. Searching and Browsing Linked Data with SWSE: the Semantic Web Search Engine. Technical Report DERI-TR-2010-07-23, 2010.
10. A. Hogan, A. Polleres, J. Umbrich, and A. Zimmermann. Some entities are more equal than others: statistical methods to consolidate Linked Data. In *Proc. of NeFoRS Workshop 2010*.
11. A. Kiryakov, D. Ognyanoff, R. Velkov, Z. Tashev, and I. Peikov. LDSR: a Reason-able View to the Web of Linked Data. In *Proc. of 7th Semantic Web Challenge*, 2009.
12. J. W. Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer-Verlag, 1987.
13. G. Meditskos and N. Bassiliades. DLEJena: A practical forward-chaining OWL 2 RL reasoner combining Jena and Pellet. *J. Web Sem.*, 8(1):89–94, 2010.
14. R. Ramakrishnan, D. Srivastava, and S. Sudarshan. Rule Ordering in Bottom-Up Fixpoint Evaluation of Logic Programs. In *Proc. of 16th VLDB*, pages 359–371, 1990.
15. D. Reynolds. OWL 2 RL in RIF. W3C Working Group Note, June 2010.
16. H. J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *J. Web Sem.*, 3:79–115, 2005.
17. J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, and H. E. Bal. OWL reasoning with WebPIE: Calculating the closure of 100 billion triples. In *Proc. of 7th ESWC, (1)*, 2010.
18. J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen. Scalable Distributed Reasoning Using MapReduce. In *Proc. of 8th ISWC*, 2009.
19. J. Weaver and J. A. Hendler. Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In *Proc. of 8th ISWC*, 2009.