

Syndication on the Web using a Description Logic Approach

Christian Halaschek-Wiener^a and Vladimir Kolovski^b

^a*Clados Management LLC, San Mateo, CA 94404, USA*¹

^b*Oracle NEDC, Nashua, New Hampshire, 03062, USA*¹

Abstract

Syndication systems on the Web have attracted vast amounts of attention in recent years. As technologies have emerged and matured, there has been a transition to more expressive syndication approaches; that is, subscribers and publishers are provided with more expressive means of describing their interests and published content, enabling more accurate information filtering. In this paper, we formalize a syndication architecture that utilizes the Web Ontology Language (OWL) and description logic reasoning for selective content dissemination. This provides finer grained control for filtering and automated reasoning for discovering implicit subscription matches, both of which are not achievable in less expressive approaches. We then address one of the main limitations with such a syndication approach, namely matching newly published information with subscription requests in an efficient and practical manner. To this end, we investigate incremental query answering for a large subset of OWL and present an approach to reduce the portion of the ontology that must be considered for query answering in the event of updates. Lastly, an evaluation of the query approach is shown, demonstrating its effectiveness for syndication purposes.

Key words: Syndication Systems, Publish/Subscribe, Description Logic, Incremental Reasoning, Query Answering

Email addresses: christian@clados.com (Christian Halaschek-Wiener), vladimir.kolovski@oracle.com (Vladimir Kolovski).

¹ The results presented in this paper are the outcome of research performed while the author was a member of the MINDSWAP research group at the University of Maryland, College Park.

1 Introduction

Web-based syndication systems have attracted a great amount of attention in recent years as the amount of streaming content on the Web has increased at dramatic rates. In typical syndication frameworks, users register their subscription requests with syndication brokers; similarly, content publishers register their feeds with syndication brokers, and it is then the broker's task to match newly published information with registered subscriptions. As technologies have emerged, there has been a transition to more expressive syndication approaches; that is publishers (and subscribers) are provided with more expressive means for describing their published content (respectively interests), allowing more accurate dissemination. This has been enabled by the maturation of technologies for sharing information on the Web and the standardization of representation languages for Web content. In particular, through the years there has been a transition from keyword based approaches to attribute-value pairs and more recently to XML. Given the lack of expressivity of XML (and XML Schema) as a knowledge modeling language, there has been interest in using the Resource Description Framework (RDF) and its accompanying schema language, RDF Schema, for syndication purposes. RDF has even been adopted as the standard representation format of RSS 1.0².

Today's syndication approaches still provide relatively weak expressive power from a modeling perspective (i.e., XML and RDF are comparatively inexpressive languages) and provide very little automated reasoning support. However, if a more expressive syndication approach with a formal semantics can be provided, many benefits can be achieved; these include a rich semantics-based mechanism for expressing subscriptions and published content allowing increased selectivity and finer control for filtering, and automated reasoning for discovering subscription matches not found using traditional syntactic syndication approaches [45].

In this work, we consider using the Web Ontology Language (OWL) for representing published content. As the semantics of a large subset of OWL is aligned with description logics (DLs), reasoning techniques for DLs can then be leveraged for matching content with subscription requests. In such an approach, the previously mentioned benefits of using a formal representation language can therefore be achieved. An additional benefit of an OWL-based syndication approach is its native Web embedding and power as a data integration language. Further, such an approach can be seen as a natural extension of existing RSS 1.0 syndication systems, as OWL can be encoded in RDF.

To demonstrate the advantages of an OWL-based syndication approach, consider the following example: suppose we are disseminating news information in the financial domain. Also suppose that a stock trader is interested in articles that *could*

² RSS 1.0 Specification: <http://web.resource.org/rss/1.0/spec>

discuss companies whose stocks are likely to become volatile; specifically, let us assume that the trader is interested in any *RiskyCompany* which the trader defines to be a *company* that has a *product* which *causes* an *infection* or *allergic reaction*. Using an XML-based approach syndication brokers can provide an XML schema that contains an element *RiskyCompany* and such companies can be declared to be this type of element. A limitation of such an approach is that publications (i.e., XML documents) must explicitly declare entities to be a *RiskyCompany*. This is because XML query languages such as XPath and XQuery only provide syntactic matching of the XML documents representing publications. If we consider an RDF-based approach, then the syndication broker can model the financial domain using RDF Schema. Therefore, additional matches can be obtained as one can logically infer that a company is a *RiskyCompany*. For example, if the domain of a property *hasProductWithAdverseEffect* is declared to be of type *RiskyCompany* and we are given that *BauschAndLomb hasProductWithAdverseEffect Renu*, then we would have a (inferred) match for the subscription; such logical inference (although simplistic) is not possible with an XML-based approach. However, in an RDF based approach, more complex logical definitions (and therefore finer-grained control) of *RiskyCompany* are not expressible.

If we now consider an OWL-based approach, such functionality is clearly provided. For example, the knowledge broker can define a *RiskyCompany* as an OWL class whose necessary and sufficient conditions for inclusion are that it be a *company* that has some product which is an *AdverseEffectProduct*; similarly, an *AdverseEffectProduct* can be defined to be any *product* that causes some *infection* or *allergic reaction*. Using an OWL approach, this can easily be represented by the OWL descriptions in Table 1³.

```

:RiskyCompany a owl:Class;
  owl:intersectionOf (
    [ a owl:Restriction; owl:onProperty :hasProduct;
      owl:someValuesFrom :AdverseEffectProduct ]
    :Company
  ) .
:AdverseEffectProduct a owl:Class;
  owl:intersectionOf (
    [ a owl:Restriction; owl:onProperty :causes;
      owl:someValuesFrom [ owl:unionOf ( :Infection :AllergicReaction ) ] ]
    :Product
  ) .
:causes a owl:ObjectProperty.
:onRecommendation a owl:ObjectProperty.

```

Table 1
Illustration of Expressivity in OWL-based Syndication.

Given this domain model, if we assume it is previously known that *BauschAndLomb* is a *company* that *has product Renu*, which is known to be a *Product*, and we receive the publication that *Renu causes* some *infection*, then standard DL rea-

³ Note that this is expressed using standard Turtle syntax (as opposed to RDF/XML) and can be easily generated in today's OWL ontology editors

soning services can be employed to automatically infer that *BauschAndLomb* is a *RiskyCompany* and thus there is a match for the subscription.

While OWL-based syndication approaches provide increased expressivity over XML and RDF, previous DL-based syndication approaches suffer from scalability issues due to the inherent complexity of DL reasoning [45,33,23]. This is an issue in domains such as the syndication of financial news because response times must be minimal as critical information must be delivered in near real time (e.g., for stock trading purposes). One of the main limitations in an OWL-based syndication approach is related to DL reasoning over changing data; this is primarily due to the static nature of existing DL reasoning techniques. In particular, the addition of information from newly published documents and data can be viewed as a change in the underlying knowledge base (KB). In current DL reasoning algorithms, reasoning on the updated KB is performed from scratch: consistency of the KB must be ensured, queries must be re-evaluated, etc.

In this paper, we formalize an OWL-based syndication framework (originally presented in [24]), which leverages DL reasoning to determine subscription matches. We then address the scalability of the DL reasoning services necessary for the syndication framework, by investigating incremental query answering over OWL KBs; we specifically present a technique for reducing the portion of the KB that must be considered as candidate query bindings after an update. This effectively allows a smaller subset of the KB to be considered for possible subscription matches. The techniques we present are applicable to conjunctive retrieval queries that can be rolled-up into a distinguished variable (discussed in the next section) and containing only simple roles (i.e., no transitive roles or super-roles of a transitive role). The approach extends our previous work [24] to support arbitrary KBs expressed in the description logic *SHI* (a large subset of OWL). Lastly, an evaluation of the incremental reasoning techniques is provided, demonstrating their effectiveness for OWL-based syndication.

2 Preliminaries

In this section, we briefly provide an overview of OWL and description logics, query answering for DL KBs, and tableau algorithms for DL reasoning.

2.1 The Web Ontology Language

The W3C-approved Web Ontology Language (OWL) is the recommended standard for formally representing content on the Web. One of the main benefits of OWL is the support for formal reasoning, as the semantics of a variety of its sub-languages

are firmly founded in description logics (a decidable fragment of First Order Logic). In particular, the sub-language OWL DL is a syntactic variant of the description logic *SHOIN* [27], with an OWL DL ontology corresponding to a *SHOIN* KB. In this work, we address a subset of *SHOIN*, namely *SHI*; therefore, we briefly introduce the syntax and semantics of *SHI*.

Let $\mathbf{C}, \mathbf{R}, \mathbf{I}$ be non-empty and pair-wise disjoint sets of *atomic concepts*, *atomic roles*, and *individuals* respectively. The set of *SHI* roles (roles, for short) is the set $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$, where R^- denotes the inverse of the atomic role R . Concepts are inductively using the following grammar:

$$C \leftarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C$$

where $A \in \mathbf{C}$, $a \in \mathbf{I}$, $C_{(i)}$ a *SHI* concept, R a role, and S a *simple* role (i.e., no transitive roles or super-roles of a transitive role)⁴. We write \top and \perp to abbreviate $C \sqcup \neg C$ and $C \sqcap \neg C$ respectively. A *role inclusion axiom* is an expression of the form $R_1 \sqsubseteq R_2$, where R_1, R_2 are roles. A *transitivity axiom* is an expression of the form $\text{Trans}(R)$, where $R \in \mathbf{R}$. An RBox \mathbf{R} is a finite set of role inclusion axioms and transitivity axioms. For C, D concepts, a *concept inclusion axiom* is an expression of the form $C \sqsubseteq D$. A TBox \mathbf{T} is a finite set of concept inclusion axioms. An ABox \mathbf{A} is a finite set of concept assertions of the form $C(a)$ (where C can be an arbitrary concept expression), role assertions of the form $R(a, b)$ and inequality (equality) assertions of the form $a \neq b$ (respectively $a = b$). A KB $\mathbf{K} = (\mathbf{T}, \mathbf{R}, \mathbf{A})$ is composed of TBox \mathbf{T} , RBox \mathbf{R} and ABox \mathbf{A} . Denote the set of individuals in KB \mathbf{K} (ABox assertion α) as $\mathbf{I}_{\mathbf{K}}$ (respectively \mathbf{I}_{α}).

An *interpretation* \mathcal{I} is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the *domain* of the interpretation, and $\cdot^{\mathcal{I}}$ is the *interpretation function*. The interpretation function assigns to $A \in \mathbf{C}$ a subset of $\Delta^{\mathcal{I}}$, to each $R \in \mathbf{R}$ a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and to each $a \in \mathbf{I}$ an element of $\Delta^{\mathcal{I}}$. The interpretation function is extended to complex roles and concepts as given in [27]. The satisfaction of a *SHI* axiom/assertion α in an interpretation \mathcal{I} , denoted $\mathcal{I} \models \alpha$ is defined as follows: (1) $\mathcal{I} \models R_1 \sqsubseteq R_2$ iff $(R_1)^{\mathcal{I}} \subseteq (R_2)^{\mathcal{I}}$; (2) $\mathcal{I} \models \text{Trans}(R)$ iff for every $a, b, c \in \Delta^{\mathcal{I}}$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ and $(b^{\mathcal{I}}, c^{\mathcal{I}}) \in R^{\mathcal{I}}$, then $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in R^{\mathcal{I}}$; (3) $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; (4) $\mathcal{I} \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$; (5) $\mathcal{I} \models R(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$; and (6) $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The interpretation \mathcal{I} is a model of a RBox \mathbf{R} (TBox \mathbf{T}) if it satisfies all the axioms in \mathbf{R} (respectively \mathbf{T}). \mathcal{I} is a model of \mathbf{A} , denoted by $\mathcal{I} \models \mathbf{A}$, if it satisfies all the assertions in \mathbf{A} . Lastly, \mathcal{I} is a model of \mathbf{K} , denoted by $\mathcal{I} \models \mathbf{K}$, iff \mathcal{I} is a model of \mathbf{T} , \mathbf{R} , and \mathbf{A} .

⁴ See [27] for a precise definition of simple roles.

2.1.1 Conjunctive Queries for Description Logics

A conjunctive query Q contains a non-empty set of concept and role atoms, $C(x)$ and $R(x, y)$ respectively, where x can be a named individual (i.e., taken from \mathbf{I}) or variable name see [29] for a precise presentation. The variable names are assumed to be typed such that each variable is either *distinguished* or *non-distinguished*; distinguished variables must be mapped to named individuals, where as the non-distinguished variables are existentially quantified. Denote the distinguished variables by $DVar(Q)$. Query answering is the task determining if Q is a logical consequence of the KB K (denoted $K \models Q$); that is, checking if for all models \mathcal{I} of K , $\mathcal{I} \models Q$. As query retrieval is addressed in this work, we briefly introduce the following notation: $(x_1, \dots, x_n) \leftarrow Q$ indicates that the variables x_1, \dots, x_n appearing in Q must be bound to individual names, therefore constituting the answer to the query (i.e., the distinguished variables). The *answer set* of a query $(x_1, \dots, x_n) \leftarrow Q$ w.r.t. to K is the set n -ary tuples defined by the following:

$$\{(a_1, \dots, a_n) \in \mathbf{I}_K^n \mid K \models Q[x_1/a_1, \dots, x_n/a_n]\}$$

where $Q[x/a]$ represents the query, Q , with all occurrences of variable x substituted by the individual name a . If a query can be partitioned into unconnected components (i.e., components that do not share variables), then they are considered independently. Without loss of generality, we assume queries are connected in the remainder of this work [17]. As retrieval queries are addressed in this work, the remainder of this discussion will only address queries of this form. Query answering can be reduced to ABox consistency checking; that is, given the query $(x) \leftarrow C(x)$ and individual a , if $K \models \text{Company}(a)$, then $K \cup \neg\text{Company}(a)$ must be inconsistent [29,43]. To extend this approach to role atoms one can transform each role atom in the query into a concept atom, which is referred to as *rolling-up* the query [29,43]. This rolling-up process is often enabled via the use of nominals, however, many DLs do not support the use of nominals; for example the DL *SHIF* (i.e., OWL Lite) does not include such expressivity. There is a well known workaround, in which the use of nominal can be simulated by substituting each nominal in the rolled-up query concept with a new concept name that does not occur in the knowledge base [29,43]. Additionally, an assertion is added to ensure that each individual instantiates its representative concept. The basic approach to support retrieval queries is to roll up the query and then to iterate over possible substitutions of named individuals for distinguished variables and perform the necessary consistency checks.

2.2 Tableau Algorithms

DL tableau-based algorithms decide the consistency of an ABox A with respect to a TBox T and RBox R by trying to construct (an abstraction of) a common

model for \mathbf{A} , \mathbf{T} , and \mathbf{R} , called a *completion graph* [27]. Each node in the graph represents an individual that is labeled with a set of concepts that it satisfies (in the particular model). Formally, a completion graph for an ABox \mathbf{A} with respect to \mathbf{T} is a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \neq)$. Each node $x \in \mathcal{V}$ is labeled with a set of concepts $\mathcal{L}(x)$, and each edge $e = \langle x, y \rangle$ with a set $\mathcal{L}(e)$ of role names. The binary predicate \neq is used for recording inequalities between nodes. This graph is constructed by repeatedly applying a set of tableau *expansion rules*, adding new concept labels and edges to the graph when necessary. This process continues until the tableau is fully expanded and no additional rules can be applied or a clash occurs. In *SHI*, a node, x , contains a clash if a contradiction exists in its label; that is $\{C, \neg C\} \subseteq \mathcal{L}(x)$. Denote a clash in $\mathcal{L}(x)$ by $(x, \neg C, C)$. It is noted that the tableau algorithm can be saturated such that all possible completion graphs of a KB are found (corresponding to all models). Denote by $Comp(\mathbf{K})$ the set of all complete and clash-free completion graphs of \mathbf{K} . When referring to the node in a completion graph that corresponds to named individual $a \in \mathbf{I}_{\mathbf{K}}$, we will denote the node by x_a and refer to it as a *root node*.

We note that reasoning with a general TBox \mathbf{T} and role hierarchy \mathcal{R} can be reduced to only reasoning with \mathcal{R} . This is because the TBox can be *internalized* into a single concept that is added to all individuals [27]. This process is briefly introduced⁵, as it will be referred to later. First, all TBox axioms in the KB are transformed into a single concept as follows:

$$C_T = \prod_{C_i \sqsubseteq D_i \in \mathbf{K}} \neg C_i \sqcup D_i.$$

Following this, a transitive role U is introduced that does not occur in the KB, and the role hierarchy for the KB is extended such that U is a transitive super-role of all roles occurring in the KB; that is, the role hierarchy is extended as follows: $\mathcal{R}_U = \mathcal{R} \cup \{R \sqsubseteq U, \text{Inv}(R) \sqsubseteq U \mid R \text{ occurs in } \mathbf{K}\}$. Given this, it has been shown that the consistency of the KB can be reduced to checking the consistency of simply the ABox w.r.t the role hierarchy \mathcal{R}_U by extending the ABox with $(C_T \sqcap \forall U.C_T)(a)$ for all named individuals $a \in \mathbf{I}_{\mathbf{K}}$ [27]. Given a concept D , let $\text{clos}(D)$ denote the smallest set of concepts containing D that is closed under sub-concepts in negation normal form (NNF)⁶. Then given KB \mathbf{K} , let $\text{clos}(\mathbf{K})$ denote the union of $\text{clos}(C_T)$ and all $\text{clos}(D)$ for each concept assertion $D(a) \in \mathbf{K}$; that is $\text{clos}(\mathbf{K}) = \text{clos}(C_T) \cup \bigcup_{D(a) \in \mathbf{K}} \text{clos}(D)$.

⁵ See [27] for a more detailed discussion.

⁶ See [27] for a discussion regarding this form.

3 Syndication Framework

In this section, we formally define the DL-based syndication framework proposed in this work. In the framework, a *publication* is a set of ABox assertions (which correspond to a set of OWL instance assertions). A publication is also associated with a number of time units in which the publication is valid; after the specified time units have passed, the publication is discarded from the syndication broker's KB (discussed below). Additionally, a boolean value is associated with a publication, denoting if the assertions should be added (or retracted) to (respectively from) the brokers KB. Retractions are supported as in many realistic syndication applications, revisions to previous publications are sometimes necessary; such a revision can be viewed as a deletion followed by an addition.

Definition 1 (*Publication*) A publication P is defined as a tuple (β, t, v) , where β is a set of DL ABox assertions that expire after t time units ($t > 0$), and v is a boolean value that is true in the event of an addition and false for retractions.

Given a publication P , we denote the set of ABox assertions as $P(\beta)$, the expiration time as $P(t)$, and the boolean addition/retraction value as $P(v)$. Next, a *publisher* is composed of a set of publications and an unique identifier:

Definition 2 (*Publisher*) A publisher Pub is defined to be a pair (p, i) , where p is a set of publications and i is an unique identifier.

We use $\text{Pub}(p)$ and $\text{pub}(i)$ to denote a publisher's set of publications and identifier respectively.

The main component of a subscription is a conjunctive ABox query, which represents the subscribers interests. In the framework, subscriptions are represented as retrieval queries; this assumption is made as it provides various match types (discussed soon), and in many real world applications investigated in OWL and DL literature, including publish/subscribe applications [23], queries typically have some number of distinguished variables. A subscription is also composed of the number of time units that the subscription is valid. Intuitively, the subscription query can be thought of as a *continuous conjunctive query* (defined below) that should be evaluated for the specified number of time units. Therefore, the query is issued once over a changing KB and the answer set of the query is continuously updated as the ABox changes.

Definition 3 (*Continuous Conjunctive Retrieval Query*) Given a continuous conjunction ABox retrieval query Q_c with n distinguished variables (i.e., $DVar(Q_c) = \{d_1, \dots, d_n\}$), define the answers of K at time t , denoted K_t , to Q_c to be those n -tuples $(a_1, \dots, a_n) \in \mathbf{I}_K^n$, such that the following holds:

$$K_t \models Q_c[d_1/a_1, \dots, d_n/a_n]$$

When referring to a continuous conjunctive retrieval query, “continuous” or “incremental” query will be used. Given a continuous query, denote the set of answer tuples at time t by $Q_c(t)$. A *subscription* is assumed to be composed of a continuous query, in addition to a number of time units which the query should be evaluated:

Definition 4 (*Subscription*) A subscription S is defined as a pair (Q_c, t) , where Q_c is a continuous query that is evaluated for t time units ($t > 0$).

The continuous query of a subscription is denoted as $S(Q_c)$ and the expiration time is denoted as $S(t)$. Next, a *subscriber* is introduced and intuitively is composed of a set of subscriptions and an unique identifier:

Definition 5 (*Subscriber*) A subscriber Sub is defined to be a pair (s, i) , where s is a set of subscriptions and i is an unique identifier.

Similar to publishers and subscriptions, $Sub(s)$ and $Sub(i)$ denotes a subscriber’s set of subscriptions and identifier respectively. A *syndication broker* maintains a local DL KB, in which newly published information is integrated. In the framework, this KB can initially contain a TBox and ABox providing background domain information. It is assumed that ABox assertions in publications utilize the terminology defined in the TBox of the broker and thus the broker’s TBox is fixed. Lastly, the syndication broker maintains the currently registered subscribers, which have associated subscriptions, and publishers. This is formally defined as follows:

Definition 6 (*Syndication Broker*) A syndication broker B is defined as a triple (S, P, K) , where S is a set of subscribers, P is a set of publishers, and K is the broker’s local DL KB.

A syndication broker’s subscribers, publishers, and KB are denoted as $B(S)$, $B(P)$, and $B(K)$ respectively. If it is clear from the context of the discussion, the broker’s KB (resp. KB at time t) will simply be referred to as K (resp. K_t). Additionally, the notation $B(K_p)$ will be used to denote the set of ABox assertions present in the broker’s KB that are included in a non-expired publication.

After a new publication is received, it is the broker’s task to determine the subscribers for which this new information is relevant. Prior to doing this, the new publications must be integrated in the broker’s KB. Unfortunately, there have been recent negative results found when trying to apply leading theories for handling updates in DL KBs. For example, the minimal model change update semantics [48] is not representable in OWL DL [34] and the postulates imposed by the AGM belief revision model [2] cannot be satisfied for OWL DL [16]. Given this, in this framework a syntactic change/update of ABox assertions is adopted, referred to as *syntactic updates*. Intuitively, syntactic updates can be described as an update in which all new ABox assertions are directly added (or removed) to the asserted (base) axioms.

Definition 7 (*Syntactic Updates*) Let \mathbf{A} be the ABox of DL KB \mathbf{K} . Then under syntactic updates, updating \mathbf{K} with an ABox addition (resp. deletion) β , written as $\mathbf{K} + \beta$ (resp. $\mathbf{K} - \beta$), results in an updated set of ABox axioms \mathbf{A}' such that $\mathbf{A}' = \mathbf{A} \cup \{\beta\}$ (resp. $\mathbf{A}' = \mathbf{A} \setminus \{\beta\}$).

If the update type is clear from the context of the discussion, $\mathbf{K} \oplus \beta$ will simply be used to denote the syntactic update of \mathbf{K} with β . Under this type of updates, the broker's KB can become inconsistent in the event of an addition; in this case, some action to regain consistency must be taken in order to allow reasoning to be performed. Therefore, in the remainder of this paper, it is assumed that if a new publication causes the broker's KB to become inconsistent, that publication is rejected; later, we discuss more advanced techniques to regain consistency.

Let us now consider subscription matches. As information is published from multiple publishers and can remain valid in the broker's local KB for varying time periods, a match for a subscription can be a composition of the information from multiple publications. Further, the broker's KB could additionally contain background knowledge which can attribute to subscription matches as well. To our knowledge, recent approaches have not investigated such functionality (e.g., [11,40,45,33]); rather, only information from individually published documents form a match for a given subscription. Such a capability is beneficial, as information can be considered collectively and form matches not found otherwise (examples are discussed later). Note that it is assumed that deletion publications do not remove initial ABox assertions in the broker's KB that correspond to background information.

A distinction between two types of subscription matches is now made, namely *information* and *publication* matches. Intuitively, an *information match* refers to the individuals bound to the distinguished variables of a continuous query representing a subscription. This type of match aligns with recent work in XML-based syndication literature, in which the actual information is filtered and the query answers are returned to the user (e.g., [32]). In contrast, a *publication match* refers to the collection of publications that satisfy a subscription; that is, given an information match for a registered subscription, it is all minimal sets of publications that cause this match to occur. This aligns with the task of selective content-based filtering of publications (e.g., [11]). The distinction between these two match types is made as the type of match required is application dependent; for example, in OWL-based syndication of news feeds, publication matches are needed. In contrast, in the financial domain, analysts are generally interested with the actual information rather than the documents themselves. We now define an *information match* as follows:

Definition 8 (*Information Match*) Define a n -tuple of individuals $(a_1, \dots, a_n) \in \mathbf{I}_{\mathbf{K}_t}^n$ to be an information match, denoted \mathcal{M}_t , at broker \mathbf{B} for subscription \mathbf{S} at time t , if the following condition holds:

$$\mathbf{B}(\mathbf{K}_t) \models \mathbf{S}(Q_c)[x_1/a_1, \dots, x_n/a_n]$$

Due to the fact that publications can persist at the syndication broker for varying time periods, answer tuples may remain valid for varying time periods as well. Given this, there are various ways in which the broker could maintain these answers and notify subscribers. For example, the broker could maintain a list of all current bindings and only forward new information matches. However, this will have some ramifications with respect to the space that it takes to store the answer sets. In contrast, in some applications it may be better to pass all current bindings to the subscriber; however, yet again, there are performance impacts due to such an approach related to the transmission cost of transferring all bindings (including those already transferred) to a subscriber. Given the fact that this is application dependent, in the formalization it is not dictated how an actual instantiation of this framework should proceed with respect to this issue; rather it is left to the individuals deploying such a framework.

Related to this issue is that a previous information match may be invalidated in the event of a retraction publication (or the expiration of a publication). Once again, there are various notification strategies that can be adopted. Specifically, a subscriber could be notified if a previous information match is invalidated due to a retraction publication; in contrast, such a notification may not be necessary in some scenarios. Therefore, such a decision is not imposed here. When discussing examples, specific decisions regarding these issues will be made if it is not clear from the context of the discussion.

Given an information match, additional computation is needed to derive *all* the minimal sets of publications responsible for the entailment. Clearly, in the event of a new information match for a subscription, the most recent (addition) publication which is received at the broker contributes to a publication match. However, we must determine the other publications which contribute to the match. For this purpose, the notion of *minimal justifications* for an entailment in DLs is utilized. This topic has been formally investigated in literature (e.g., [31]) and techniques have been developed to solve this problem. For purpose of this work, we leverage work on minimal justifications [31], which are defined below:

Definition 9 (*Minimal Justification*) Let $K \models \alpha$, where α is a DL axiom and K a DL KB. A fragment $K' \subseteq K$ is a *minimal justification* for α in K if $K' \models \alpha$ and $K'' \not\models \alpha$ for every $K'' \subset K'$. Denote the set of minimal justifications for $K \models \alpha$ as $\text{Just}(K, \alpha)$.

Given this, we present the definition of a *publication match*:

Definition 10 (*Publication Match*) Let a n -tuple of individuals $(a_1, \dots, a_n) \in \mathbf{I}_{K_t}^n$ be an information match, \mathcal{M}_t , at broker B at time t for subscription S , where $S(Q_c)$ has n distinguished variables (i.e., $DVar(S(Q_c)) = \{d_1, \dots, d_n\}$). Additionally, let J be the set of minimal justifications for \mathcal{M}_t such that:

$$J = \text{Just}(K_t, S(Q_c)[d_1/a_1, \dots, d_n/a_n])$$

Define a set of publications P to be a publication match, denoted M_P , at broker B for subscription S at time t if there exists $j \in J$ such that the following holds:

- for all $P \in P$, there exists some $\alpha \in j$ such that $\alpha \in P(\beta)$ and
- for all $\alpha \in j$ one of the following holds:
 - there exists some $P \in P$ and $\alpha \in P(\beta)$ or
 - $\alpha \notin B(K_P)$

As in the case for information matches, there are various ways to proceed related to the manner in which subscribers should be notified about publications matches (e.g., in the event of a retraction publication, subscribers could be notified about invalidated publication matches). Again, due to the application dependence of such a decision, a choice is not imposed here.

We conclude this section with a brief example demonstrating matches and the framework in general. Let us assume that a syndication broker B is aware of two subscribers, S_1 and S_2 , and two publishers, P_1 and P_2 . Additionally, assume that the broker contains existing background information in its knowledge base, including the axioms defined previously in Table 1, in addition to the type assertions that *BauschAndLomb* is a *Company* and *FusariumEyeInfection* is an *Infection*. Assume that S_1 is interested in information about risky companies. Given this interest, S_1 registers a non-expiring subscription with the broker that is composed of a continuous query for all individuals of type *RiskyCompany*. On the other hand, assume that S_2 is interested in products that have some adverse effect; these subscriptions are represented as follows, where ∞ indicates that the subscription does not expire:

$$\begin{aligned} ((x) \leftarrow \text{RiskyCompany}(x), \infty) &\in S_1(s) \\ ((x) \leftarrow \text{AdverseEffectProduct}(x), \infty) &\in S_2(s) \end{aligned}$$

Next, suppose that P_1 publishes an addition publication that expires in 24 hours which contains the assertions that *BauschAndLomb* has a product named *Renu* and that *Renu* is a *Product*. In addition, assume that P_2 publishes an addition publication that also expires in 24 hours, however it contains the assertion that *Renu* causes the *FusariumEyeInfection*. These publications are formally represented as follows, where *24h* indicates that the publications expire in 24 hours:

$$\begin{aligned} P_{P_1} &= (\{ \text{hasProduct}(\text{BauschAndLomb}, \text{Renu}), \text{Product}(\text{Renu}) \}, 24h, \text{true}, P_1) \\ P_{P_2} &= (\{ \text{causes}(\text{Renu}, \text{FusariumEyeInfection}) \}, 24h, \text{true}, P_2) \end{aligned}$$

Assume that P_{P_1} and P_{P_2} arrive at the broker at time 1 and 2 respectively. When P_{P_1} arrives at the broker, $P_{P_1}(\beta)$ is integrated into $B(K)$, resulting in an updated broker KB K' . At this time the individual *BauschAndLomb* will not satisfy the subscription, as it cannot be inferred that *Renu* is an *AdverseEffectProduct*; therefore, there will not be a match for S_1 or S_2 at time 1. However, when P_{P_2} is published at time 2 and integrated into K' , there will be a new information matches for both registered subscriptions, as the broker's KB now entails that *Renu* is an *AdverseEffectProduct*

and that *BauschAndLomb* is a *RiskyCompany*. This is fairly straightforward given the domain model and assertions in the syndication broker’s KB. There is additionally a composite publication match $\{P_{P_1}, P_{P_2}\}$ for both subscriptions. We only consider the subscription from S_1 , as the second follows in a similar manner; first, we must determine the justifications for the entailment that *BauschAndLomb* is a *RiskyCompany*. In this case, there is only one justification: $\{Company(BauschAndLomb), Product(Renu), hasProduct(BauschAndLomb, Renu), causes(Renu, FusariumEyeInfection)\}$ ⁷. Collectively P_{P_1} and P_{P_2} form a publication match, as P_{P_1} contains the assertions *hasProduct(BauschAndLomb, Renu)* and *Product(Renu)*, while P_{P_2} contains *causes(Renu, FusariumEyeInfection)*. Note that *Company(BauschAndLomb)* was originally contained in the brokers KB independent of any publications (i.e., it is background information); therefore, $\{P_{P_1}, P_{P_2}\}$ satisfies the definition of a publication match.

As discussed earlier, the main limitation in the proposed syndication framework is related to DL reasoning through incremental changes to the underlying KB. Therefore, the remainder of this paper addresses the two required reasoning services within the syndication framework; namely consistency checking and query answering through updates.

4 Incremental Consistency Checking

After newly published information is integrated in the broker’s KB, consistency must be re-checked. With large ABoxes, checking consistency introduces substantial overhead. In the case of syndication, this problem is compounded as the broker’s KB will become substantially large because the KB can contain permanent domain knowledge, as well as publications that remain valid for substantial time periods. We have recently investigated incremental consistency checking in OWL KBs. In [25,26] we present an approach for incrementally updating tableau completion graphs under syntactic ABox updates in the description logics *SHIQ* and *SHOQ* [25,26], which encompass a large portion of OWL DL. To support addition updates the update algorithm adds new components (edge, nodes, or labels) introduced by the update to a (cached) completion graph from the consistency check prior to the update; after this, standard tableau completion rules are re-fired to ensure that the model is complete. In order to support deletions, we extend work on *axiom tracing* [31,4] so that the dependent structures in a previous completion graph can be rolled back. In both cases, the completion graph built prior to the update is updated such that if a model exists (i.e, the KB is consistent after the update), then a new complete and clash-free completion graph will be constructed. It was observed that updates do not have a large effect on the existing completion graph; therefore, orders of magnitude performance improvements are achieved [25,26].

⁷ For ease of presentation, we have omitted TBox axioms from the justification.

5 Incremental Query Answering

After guaranteeing consistency of an updated KB, the various subscriptions registered with the broker can be (re)evaluated. One of the main issues w.r.t. to the syndication framework with current query answering algorithms is that after an update the entire knowledge base is considered when re-evaluating the query. Given this, the technique developed in this section aims to reduce the portion of the KB that must be considered as candidate answers after an update; therefore, the query only needs to be re-evaluated over a subset of the KB. The technique developed is applicable to the DL *SHI* (a large subset of OWL DL). Thus, in the remainder of this section, it is assumed all KBs are expressed in *SHI*.

Let us assume that we are given a retrieval query $(x) \leftarrow C(x)$ for some concept C and named individual a . Additionally, say that an ABox update β is integrated into the KB under syntactic updates, such that the resulting KB is consistent. The main insight underlying the approach is that the dependencies of clashes in completion graphs for the KB caused by β and $\neg C(a)$ can be exploited to provide an overestimate of the individuals that instantiate C after the update. Before formally presenting the necessary conditions for the entailment, the definition of the dependency of a node label in a completion graph is presented.

Definition 11 (*Label Dependence*) *Define a node label $l \in \mathcal{L}(x)$ to be dependent on a node label $C \in \mathcal{L}(y)$ (or node $y \in \mathcal{V}$, edge $\langle y, z \rangle \in \mathcal{E}$, or edge label $R \in \mathcal{L}(\langle y, z \rangle)$) if during the application of expansion rules to construct completion graph G , l is added to $\mathcal{L}(x)$ due to the existence of $C \in \mathcal{L}(y)$ (respectively $y \in \mathcal{V}$, $\langle y, z \rangle \in \mathcal{E}$, $R \in \mathcal{L}(\langle y, z \rangle)$).*

The notion of clash dependence is a straightforward extension of this definition; that is, if a clash $c = (x, \neg C, C)$ is observed and either $\neg C$ or C is dependent on some structure s (node, edge, or label), then the clash is said to be dependent on s . Finally, we say that a label $l \in \mathcal{L}(x)$ is dependent on an update β if β causes the addition of some structure s s.t. l is dependent on s ; note that a clash dependency on an update can be defined in a similar way. Given this, the main theorem underlying the approach developed in this section is introduced. For ease of exposition, when referring to the addition of the structure of a set of ABox assertions β to a completion graph G and then applying any sequence of the necessary expansion rules (as discussed in Section 4⁸), the terminology “adding β to G ”, denoted $G \uplus \beta$, will be used.

⁸ In this discussion, it is assumed that back-jumping is not used, as all completion graphs are maintained.

Theorem 1 Let K be a consistent \mathcal{SHI} KB, β an ABox addition (or deletion), and C some \mathcal{SHI} concept. If $K \not\models C(a)$ (respectively $K \models C(a)$) for some $a \in \mathbf{I}_K \cup \mathbf{I}_\beta$, $K + \beta \not\models \perp$, and $K + \beta \models C(a)$ (respectively $K - \beta \not\models C(a)$), then one of the following conditions is satisfied:

- (1) there exists $G \in \text{Comp}(K)$ (respectively $G \in \text{Comp}(K - \beta)$) s.t. $G \uplus (\beta \cup \{\neg C(a)\})$ results in a clash c that is dependent on both $\neg C(a)$ and β
- (2) there exists the same node x with $D_1 \sqcup D_2 \in \mathcal{L}(x)$ in $\{G_1, G_2\} \subseteq \text{Comp}(K)$ (respectively $\{G_1, G_2\} \subseteq \text{Comp}(K - \beta)$) such that:
 - $G_1 \uplus (\beta \cup \{\neg C(a)\})$ results in a clash that is independent of $\neg C(a)$ and is dependent on both β and $D_1 \sqcup D_2 \in \mathcal{L}(x)$
 - $G_2 \uplus (\beta \cup \{\neg C(a)\})$ results in a clash that is independent of β and is dependent on both $\neg C(a)$ and $D_1 \sqcup D_2 \in \mathcal{L}(x)$

Proof Consider addition updates. This case will be shown by contradiction. Assume that 1) $K \not\models C(a)$, $K + \beta \not\models \perp$, $K + \beta \models C(a)$, 2) there does not exist $G \in \text{Comp}(K)$ s.t. the first condition of the theorem holds, and 3) there does not exist the same node x with $D_1 \sqcup D_2 \in \mathcal{L}(x)$ in $\{G_1, G_2\} \subseteq \text{Comp}(K)$ s.t. the second condition of the theorem holds. First note that a) $\text{Comp}(K + \{\neg C(a)\}) \neq \emptyset$, as $K \not\models C(a)$, b) $\text{Comp}(K + \beta) \neq \emptyset$, as it is assumed $K + \beta \not\models \perp$, c) $\text{Comp}(K + \{\neg C(a)\} + \beta) = \emptyset$, as $K + \beta \models C(a)$, and d) the additional expansion rule applications caused by adding β and $\neg C(a)$ to each $G \in \text{Comp}(K)$ must cause a clash; this follows from the completeness of the incremental consistency checking algorithm (shown in Theorem 2 of [26]) & property c. Next, assumption 2 implies that every clash observed when updating each $G \in \text{Comp}(K)$ with $\neg C(a)$ and β is not dependent on both $\neg C(a)$ and β . This implies that each observed clash c must be dependent on $\neg C(a)$ or β (but not both), as well as some non-deterministic choice (i.e., some $D_1 \sqcup D_2 \in \mathcal{L}(x)$ for some $x \in \mathcal{V}$); otherwise, $K \models C(a)$ or $K + \beta \models \perp$ must hold. Clearly, every clash observed cannot be dependent on β (or $\neg C(a)$), as this would imply $K + \beta \models \perp$ (respectively $K \models C(a)$). Next, it must be the case that for some set of clashes $\{c_1, \dots, c_n\}$ observed, any c_i , $1 \leq i \leq n$, is dependent on the same non-deterministic choice as any c_j , $i \neq j$; this is a consequence of the previous properties and the fact that each clash observed is dependent on a non-deterministic choice. It suffices to show that for some set of observed clashes $\{c_1, \dots, c_n\}$ that are dependent on the same non-deterministic choice, there exists c_i , $1 \leq i \leq n$, that is dependent on β and there exists c_j , $i \neq j$, that is dependent on $\neg C(a)$. This follows easily as if this were not the case then either $K + \beta \models \perp$ or $K \models C(a)$. Note that the only cause of non-determinism in the \mathcal{SHI} tableau algorithm are disjunctions; thus c_i and c_j must be dependent on the same disjunction. This implies that there exists $\{G_1, G_2\} \subseteq \text{Comp}(K)$ with the same node x s.t. c_i and c_j are dependent on $D_1 \sqcup D_2 \in \mathcal{L}(x)$, and c_i and c_j are dependent on β and $\neg C(a)$ respectively. Thus, we have arrived at a contradiction. Due to the assumption that $K \models C(a)$ and $K - \alpha \not\models C(a)$, it is the case that $(K - \alpha) + \alpha \models C(a)$. Therefore, the case for deletions is a consequence of the case for additions. \square

It is important to reiterate that in condition 2 of the theorem, the node x in G_1

and G_2 corresponds to the same node (which corresponds to either a named or existential individual). Note that the case in which x corresponds to an existential is not problematic to maintain, as nodes are not merged and all completion graphs are maintained; therefore, when a disjunction is encountered on an existential and various new completion graphs are constructed, the correspondences between the existential node x in each of the completion graphs can trivially be determined.

Intuitively, the first condition of Theorem 1 states that for a named individual a to instantiate a concept after an addition, then some clash observed when incrementally updating a completion graph for the original KB with β and $\neg C(a)$ will be dependent on structures from both β and $\neg C(a)$. On the other hand, the second condition states that $\neg C(a)$ and β will cause clashes in different completion graphs that are dependent on the same non-deterministic choice (i.e., a disjunction label). Analogous statements can be made for deletions. Thus, given a retrieval query composed of a single DL concept, if all completion graphs for the KB are maintained through updates and the two conditions are checked, then the detection of new candidate bindings (respectively invalidated bindings for deletions) for a given query can be accomplished; then, only this subset of the individuals would have to actually be checked for the entailment. The main insight is that if this technique (or an overestimate of it) can be accomplished in a practical manner, then this set of candidates may be a small subset of the original KB, thereby decreasing the overhead of re-evaluating queries given an ABox update. In the remainder of this section, we briefly discuss a naïve approach which exploits Theorem 1 for this task. Then, we discuss how to make the technique practical.

Let us first consider addition updates; to take into account the first condition of Theorem 1 for additions, one can update all completion graphs for the initial KB with β and track the label dependencies for β . Following this, one can update the resulting complete and clash-free completion graphs with $\neg C(a)$ and determine which clashes are dependent on both β and $\neg C(a)$. This is sufficient due to the fact that there is not an expansion rule application ordering imposed for *SHI* [28] and condition 1 of Theorem 1 states the clash must be dependent on both β and $\neg C(a)$ when applying any sequence of the necessary expansion rules⁹. An additional observation related to this is made; for a clash to be dependent on both β and $\neg C(a)$, then after adding β to all $G \in \text{Comp}(\mathbb{K})$ resulting in the set of complete and clash-free completion graphs $G_{\mathbb{K}+\beta}$, it must be the case that when adding $\neg C(a)$ to some $G' \in G_{\mathbb{K}+\beta}$, a root node that had a node label, edge, or edge label added due to β must have a label added due $\neg C(a)$. This is a consequence of the previous observation and the tree-like model property of *SHI*, which intuitively states that the completion graph will be a forest of trees rooted at nodes corresponding to named

⁹ Note, however, there could be a case in which there is an immediate clash as a result of adding the structures of β and $\neg C(a)$ to some $G \in \text{Comp}(\mathbb{K})$ prior to applying any expansion rules. Using this approach, such a case would not be detected; however this case can trivially be covered by inspecting β and $\neg C(a)$.

individuals. Therefore, one can easily determine an over-estimate of the individuals that could satisfy condition 1 by first updating all $G \in \text{Comp}(\mathbf{K})$ with β , while tracking the root nodes N with a node label, edge, or edge label change. Then, the negated query concept, $\neg C$, can be to $\mathcal{L}(x_a)$ ¹⁰ in each updated completion graph, and for the condition to be satisfied for a , a label must be added to some $n \in N$ due to the addition of $\neg C$ to $\mathcal{L}(x_a)$.

A straightforward observation is made regarding the second claim for Theorem 1 as well. In particular, when updating each $G \in \text{Comp}(\mathbf{K})$ with β , all clashes dependent on β that are independent of $\neg C$ will be observed. Therefore, if the dependencies of labels on disjunctions are tracked during the tableau algorithm, one can easily determine the individuals that satisfy the second condition by adding $\neg C$ to each individual and checking if this causes a clash that is dependent on some disjunction that contributed to a clash when updating each $G \in \text{Comp}(\mathbf{K})$ with β . An over-estimate of this approach can be provided in an analogous manner as the previous case. Such a disjunction dependency tracking function can easily be maintained in a similar manner to the axiom tracing technique discussed in Section 4. In particular, when a disjunction is added to a node in a completion graph, the function will be updated with the label addition events that are a result of the disjunction. As this can be accomplished in an analogous manner as axiom tracing, this function is simply assumed.

Now consider deletion updates; by additionally leveraging the axiom tracing function discussed in Section 4, a similar technique to the approach just discussed for addition updates can be used. Specifically, due to the completeness of the axiom tracing function, the complete and clash-free completion graphs for $\mathbf{K} - \beta$ can be obtained by rolling-back the change events in each $G \in \text{Comp}(\mathbf{K})$ that are dependent on some $\alpha \in \beta$ and then applying the necessary expansion rules (see [26] for more details). Thus, the deletions can be supported in the same manner as additions.

It is clear that adopting such a naïve approach to determine the candidate individuals will impose substantial overhead, and likely be worse than simply running the query from scratch. This is because one would have to perform such a check for each named individual in all completion graphs, which would clearly be ineffective. Further, maintaining all completion graphs for the KB is itself impractical due to the potential exponential number of completion graphs. Given this, in the remainder of this paper, a more practical approach to exploit Theorem 1 is presented. First, the issue of adding the negated query concept to all named individuals is addressed in Section 5.1. Following this, we develop a technique to avoid maintaining all completion graphs for the KB.

In the techniques developed in the remaining sections of this paper some restrictions and assumptions are imposed on the queries supported. First, it must be the

¹⁰ Given a named individuals a , we denote by x_a the root node corresponding to a .

case that the query can be rolled-up into a distinguished variable. This implies that the rolling-up technique must be applicable to the query and that it must contain at least one distinguished variable (implying it is a retrieval query). While this is a restriction of the approach in general, it is noted that the requirement that the query is a retrieval query does not directly impact the utility of the technique for the purpose of the syndication framework, as subscriptions are in fact retrieval queries. The second restriction is that role atoms in the query must be simple roles; this is necessary to ensure completeness of the proposed techniques. It is noted, that in general rolling-up arbitrarily shaped queries in the presence of transitive roles is known to be problematic [43,18]; therefore, this restriction implies that the query can in fact be rolled-up (as required in the first restriction). Lastly, an additional syntactic restriction is imposed on the queries supported, however this will be discussed shortly in the next section.

5.1 Concept Guide

In this section, an approach is presented for avoiding the addition of the negated query concept to all named individuals. The main goal behind the approach is to build a structure that can be used to determine the propagation of labels to root nodes in a completion graph due to the addition of the negated query concept. In the remainder of this section, only retrieval queries involving a single DL concept are assumed (i.e., concept retrieval queries); however, the technique introduced here is extended to complex query patterns in Section 5.3.

In order to support general TBoxes, a restriction is imposed on the queries supported in the approach. Given retrieval query for concept C (in NNF), it is assumed that if $\forall R.D \in \text{clos}(K) \cup \text{clos}(\neg C)$, then it must be the case that $\exists P.E \notin \text{clos}(\neg C)$, where $\text{Inv}(P) \sqsubseteq R$. Intuitively, given a complete and clash-free completion graph G and some named individual a , this restriction ensures that if a new edge is added to G as a result of extending G with $\neg C(a)$, then no concepts will be transferred back up this newly added edge. This effectively isolates the propagation of labels due to expansion rule applications from the addition of the negated query concept to G (we discuss the impact of this restriction in practice later). Given a KB K and concept C , we will say that C is *safe* with respect to K if C satisfies the restriction just introduced.

We now introduce the structure that is leveraged for determine the effects of adding a concept name to the label of a node in a completion graph; this is referred to as a *concept guide*. Intuitively, a concept guide is a labeled, directed graph, that is built by repeatedly inspecting the form of the concept names in node labels in the concept guide.

Definition 12 (Concept Guide) Define a concept guide \mathcal{G} to be a labeled directed graph $\mathcal{G} = (N_{\mathcal{G}}, E_{\mathcal{G}}, L_{\mathcal{G}})$. Each node $n \in N_{\mathcal{G}}$ is labeled with a non-empty set of *SHI* concepts, and each edge $(n, m) \in E_{\mathcal{G}}$ is labeled with a non-empty set of role names. Given *SHI* KB \mathcal{K} , *SHI* concept C , and $\mathcal{G} = (\emptyset, \emptyset, L)$, define the concept guide for C , denoted $\text{guide}(C)$, to be initialized with $n \in N_{\mathcal{G}}$ and $L_{\mathcal{G}}(n) \leftarrow \{C\}$. Then define the following rules to be repeatedly applied to the concept guide node labels until no further modifications can be made to \mathcal{G} :

- (1) if $\forall R.D \in L_{\mathcal{G}}(n)$ and $(n, m) \notin E_{\mathcal{G}}$ s.t. $R \in L_{\mathcal{G}}((n, m))$ and $D \in L_{\mathcal{G}}(m)$ then
 - (a) add a new node m to $N_{\mathcal{G}}$ and set $L_{\mathcal{G}}(m) \leftarrow \{D\}$
 - (b) set $E_{\mathcal{G}} \leftarrow E_{\mathcal{G}} \cup \{(n, m)\}$
 - (c) set $L_{\mathcal{G}}((n, m)) \leftarrow L_{\mathcal{G}}((n, m)) \cup \{R\}$
- (2) if $\forall R.D \in L_{\mathcal{G}}(n)$, there is some S with $\text{Trans}(S)$ and $S \sqsubseteq R$, and $\{(n, m), (m, m)\} \notin E_{\mathcal{G}}$ s.t. $R \in L_{\mathcal{G}}((n, m))$, $D \in L_{\mathcal{G}}(m)$, and $S \in L_{\mathcal{G}}((n, m))$ and $S \in L_{\mathcal{G}}((m, m))$ for all S s.t. $\text{Trans}(S)$ and $S \sqsubseteq R$ then
 - (a) add a new node m to $N_{\mathcal{G}}$ and set $L_{\mathcal{G}}(m) \leftarrow \{D\}$
 - (b) set $E_{\mathcal{G}} \leftarrow E_{\mathcal{G}} \cup \{(n, m)\} \cup \{(m, m)\}$
 - (c) set $L_{\mathcal{G}}((n, m)) \leftarrow L_{\mathcal{G}}((n, m)) \cup \{R\}$
 - (d) set $L_{\mathcal{G}}((n, m)) \leftarrow L_{\mathcal{G}}((n, m)) \cup \{S\}$ and $L_{\mathcal{G}}((m, m)) \leftarrow L_{\mathcal{G}}((m, m)) \cup \{S\}$ for all S s.t. $\text{Trans}(S)$ and $S \sqsubseteq R$
- (3) if $C_1 \sqcap C_2 \in L_{\mathcal{G}}(n)$ or $C_1 \sqcup C_2 \in L_{\mathcal{G}}(n)$ and $\{C_1, C_2\} \not\subseteq L_{\mathcal{G}}(n)$ then set $L_{\mathcal{G}}(n) \leftarrow L_{\mathcal{G}}(n) \cup \{C_1, C_2\}$

It is a straightforward consequence of the definition that the construction of the concept guide will terminate; this is implied by the conditions checked prior to performing the operations in the definition and the fact that $\text{clos}(C)$ is of finite size. To demonstrate the construction of a concept guide, let us consider the previous query from Section 5, $(x) \leftarrow (\exists \text{hasProduct} . (\exists \text{causes} . \text{Infection}))(x)$. Additionally, assume that *hasProduct* is a transitive role; observe that the negation of the query concept is $\forall \text{hasProduct} . (\forall \text{causes} . \neg \text{Infection})$. Let us consider the concept guide for the negated query concept; when constructing the concept guide, a new node will first be added and labeled with the negated query concept. Following this, the second condition of Definition 12 will be applicable to this node label. Therefore, a new node and edge will be created with labels *hasProduct* and $\forall \text{causes} . \neg \text{Infection}$ respectively. Further, because the *hasProduct* role is transitive, a self-looping edge labeled with this role will be added to the most recently added node. Lastly, due to the newly added $\forall \text{causes} . \neg \text{Infection}$ label, a new edge and node will be created and labeled with *causes* and $\neg \text{Infection}$ respectively. The resulting concept guide is depicted in Figure 1.

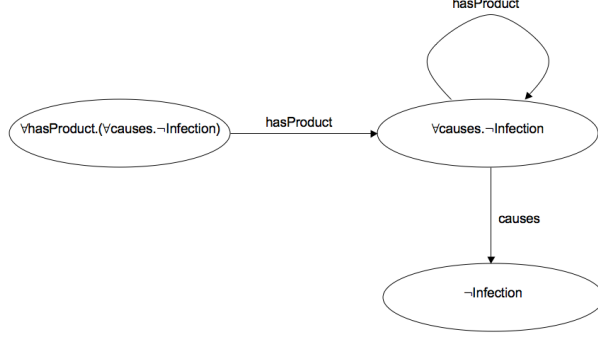


Fig. 1. Example Concept Guide.

5.1.1 Approach

In this section, we discuss how the concept guide is used to determine the propagation of labels due to an update in a completion graph. First, the notion of a *concept guide path* between two nodes in a completion graph is defined.

Definition 13 (*Concept Guide Path*): Let \mathbf{K} be a *SHI* KB, C some concept, $\mathcal{G} = \text{guide}(C)$, $G \in \text{Comp}(\mathbf{K})$, and $n, m \in N_{\mathcal{G}}$. Define there to be an n - m -concept guide path between two nodes $x, y \in \mathcal{V}$, denoted $\text{path}(n, m, x, y, \mathcal{G}, G)$, if there is a sequence of edge traversals, $\langle x_1, y_1 \rangle, \dots, \langle x_k, y_k \rangle \in \mathcal{E}$ and $(n_1, m_1), \dots, (n_k, m_k) \in E_{\mathcal{G}}$, such that the following holds:

- (1) $x_1 = x$ and $n_1 = n$ (i.e., the path starts at $n \in N_{\mathcal{G}}$ and $x \in \mathcal{V}$)
- (2) for each edge traversal, $\langle x_i, y_i \rangle \in \mathcal{E}$, and corresponding edge traversal, $(n_i, m_i) \in E_{\mathcal{G}}$, it is the case that for some $R \in L_{\mathcal{G}}((n_i, m_i))$, x_i has R -neighbor y_i
 - (a) if some node $z \in \mathcal{V}$ is blocked by $w \in \mathcal{V}$, then an edge $\langle v, w \rangle$ with $\mathcal{L}(\langle v, w \rangle) \leftarrow \mathcal{L}(\langle v, z \rangle)$, where v is the predecessor of z , is also considered for traversal
- (3) $y_k = y$ and $m_k = m$ (i.e., the path ends at $m \in N_{\mathcal{G}}$ and $y \in \mathcal{V}$)

Condition 2a is necessary due to blocking conditions utilized during the creation of *SHI* completion graphs, as cyclic models can occur. Condition 2a overcomes the problematic case where there *would* be a path in the completion graph that satisfies the constraints imposed by the concept guide, yet due to blocking, the path does not explicitly exist. Assuming the restrictions imposed on the form of C , it can be shown that if adding a concept $\neg C$ to root node x_a in a complete and clash-free completion graph causes a label to be added to a root node x_b , then there is a concept guide path that starts at the concept guide node labeled with $\neg C$ between the two nodes.

Theorem 2 Let \mathbf{K} be a *SHI* KB, $G \in \text{Comp}(\mathbf{K})$, C be a *SHI* concept that is safe with respect to \mathbf{K} , and $\mathcal{G} = \text{guide}(\neg C)$. If adding $\neg C(a)$, $a \in \mathbf{I}_{\mathbf{K}}$, to G causes a concept name to be added to $\mathcal{L}(x_b)$, $b \in \mathbf{I}_{\mathbf{K}}$, then there is a concept guide path $\text{path}(n, m, x_a, x_b, \mathcal{G}, G)$ for some $n, m \in N_{\mathcal{G}}$ s.t. $\neg C \in \mathcal{L}_{\mathcal{G}}(n)$.

The proof of this theorem is omitted, however it can be found in Appendix A.2.2 of [26]. Theorem 2 implies that the concept guide can be used to avoid adding $\neg C$ to all individuals in the KB to take into account the first condition of Theorem 1 for additions. This is because, one can maintain the root nodes x with a node label, edge, or edge label change due to β , and then search for all nodes that satisfy the concept guide path relation involving x . Similarly, the second condition of the theorem can be taken into account; that is, if the disjunction dependencies of labels are tracked during the tableau algorithm, then one can easily determine the root nodes N which have a label that is dependent on one of the disjunctions that contributed to clash observed due to β . Thus, the concept guide can then be used to determine the root nodes $n \in N$ reachable due to $\neg C$. Given the discussion presented earlier, it is clear that deletion updates can be handled in a similar manner.

Next, a variety of notation is introduced; given ABox addition β , $G \in \text{Comp}(\mathbb{K})$ and G' the result of $G \uplus \beta$ (containing a clash or clash-free), denote by $\text{Dep}(\beta, G, G')$ the set of named individuals whose corresponding root nodes have a node label, or incoming/outgoing edge or edge label changed when constructing G' . Further, denote by $\text{Dis}_{j\mathbb{K}}(D_1 \sqcup D_2, x)$ the set of named individuals whose corresponding root nodes have a label that is dependent on disjunction $D_1 \sqcup D_2 \in \mathcal{L}(x)$ in some completion graph $G \in \text{Comp}(\mathbb{K})$. Given this, we define the set of *concept candidates*, which intuitively is an over-estimate of the individuals which instantiate (or no longer instantiate) a concept after an addition (respectively deletion).

Definition 14 (*Concept Candidates*): Let \mathbb{K} be a *SHI* KB, β an ABox addition (or deletion), C be a *SHI* concept that is safe with respect to \mathbb{K} and β , and $\mathcal{G} = \text{guide}(\neg C)$. Then define the concept candidates, denoted $\text{CC}(\mathbb{K}, C, \beta)$, to be the set of all named individuals $a \in \mathbf{I}_{\mathbb{K}} \cup \mathbf{I}_{\beta}$ such that adding β to some $G \in \text{Comp}(\mathbb{K})$ (respectively $G \in \text{Comp}(\mathbb{K} - \beta)$) results in G' and one of the following conditions is satisfied for some $n, m \in N_{\mathcal{G}}$, where $\neg C \in \mathcal{L}_{\mathcal{G}}(n)$. :

- (1) $a \in \mathbf{I}_{\beta}$
- (2) G' clash-free, $b \in \text{Dep}(\beta, G, G')$ and there is a concept guide path $\text{path}(n, m, x_a, x_b, \mathcal{G}, G')$
- (3) a clash is observed that is dependent on $D_1 \sqcup D_2 \in \mathcal{L}(y)$ and for some $b \in \text{Dis}_{j\mathbb{K}}(D_1 \sqcup D_2, y)$ (respectively $b \in \text{Dis}_{j\mathbb{K}-\beta}(D_1 \sqcup D_2, y)$) or $b \in \text{Dep}(\beta, G, G')$ there is a concept guide path $\text{path}(n, m, x_a, x_b, \mathcal{G}, G'')$ in some $G'' \in \text{Comp}(\mathbb{K}) \setminus G$

Theorem 3 implies the correctness of the approach using the concept guide for determining the candidate new (respectively invalidated) bindings for a retrieval query consisting of a *SHI* concept.

Theorem 3 Let \mathbb{K} be a *SHI* KB, β an ABox addition (or deletion), C be a *SHI* concept that is safe with respect to \mathbb{K} and β , and $\mathcal{G} = \text{guide}(\neg C)$. If for some $a \in \mathbf{I}_{\mathbb{K}} \cup \mathbf{I}_{\beta}$, $\mathbb{K} \not\models C(a)$ and $\mathbb{K} + \beta \models C(a)$ (respectively $\mathbb{K} \models C(a)$ and $\mathbb{K} - \beta \not\models C(a)$), then $a \in \text{CC}(\mathbb{K}, C, \beta)$

The proof of this theorem is omitted, however it can be found in Appendix A.2.3 of [26]. We conclude this section with a brief discussion regarding the approach. First, the construction of the concept guide assumes the standard *SHI* tableau expansion rules [28]. In many DL reasoning systems, a variety of optimizations are utilized, some of which introduce additional tableau expansion rules (e.g., unfolding rule, domain/range rule [44]). It is noted that the approach can easily be extended to take into account the unfolding and domain/range expansion rules via a simple extension to the definition of the concept guide (see [26] for a more detailed discussion). Lastly, a brief comment is in order regarding the impact of this restriction in practice. To investigate this, we gathered 460 of the OWL ontologies¹¹ used during a recent survey of the publicly available OWL ontologies available on the Web [47] and investigated the actual impact of the restriction. Each concept in the ontology was selected and tested to see if it violated the restrictions imposed by the approach¹². Of the 460 ontologies, only 4 had at least one concept that violated the restriction. This indicates that the restriction should not have a large impact when attempting to use the technique for many ontologies. Further, in the 4 violating ontologies, on average 6.8 concepts actually violated the restriction. This is promising, as the number of concepts that cannot be used in queries is very small.

5.2 Summary Completion Graph

Incrementally maintaining all completion graphs for a given KB is not practical. Further, in the presence of a reasonable degree of non-determinism in a KB, constructing all completion graphs is a very expensive process. To overcome this issue, an approach is developed in which a completion graph structure is constructed that represents a summary of the structures present in all completion graphs for the KB; this structure is referred to as a *summary completion graph*. Importantly, this structure can be utilized to locate the candidate individuals, and therefore all completion graphs for the KB do not have to be maintained.

Definition 15 (*Initial Summary Completion Graph Construction*): Let S_G be the summary completion graph for *SHI* KB \mathcal{K} constructed by applying the *SHI* tableau algorithm to \mathcal{K} , however with the following modifications:

- (1) the \sqcup -rule is replaced as follows: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
- (2) if a clash is encountered, it is ignored and the algorithm continues

¹¹ Note this is a subset of the surveyed ontologies, as a number of the ontologies were no longer available.

¹² The statistics were gathered after absorption and internalization of the general TBox. It was also assumed that the unfolding and domain/range rules were utilized; therefore, the syntactic restriction was extended to cover the concept encountered when unfolding the query concept.

Termination of the construction of the summary completion graph follows easily from the fact that the termination for the *SHI* tableau algorithm is independent of clash detection; therefore it can be shown in an identical manner as termination for the *SHI* tableau algorithm [28]. The construction of the summary completion graph proceeds in an identical manner as the regular tableau algorithm, however when the \sqcup -rule is applied, all concept names of the disjunction are added to the node label (in the same manner as the \sqcap -rule). Note that condition 2 of the definition is required, as adding all concept names from a disjunction can introduce clashes which would not occur in the different completion graphs.

We now show how the summary completion graph can be used to avoid maintaining all completion graphs for the purpose of exploiting Theorem 1. This is accomplished by showing that an overestimate of the concept candidates introduced in Definition 14 can be determined by simply using the summary completion graph. Clearly, the first condition of the definition is trivial; therefore, only conditions 2 and 3 are addressed. In the following discussion, we only address addition updates, as the approach for deletions follows in a similar manner and will be addressed later.

For the second condition of Definition 14 to be taken into account using the summary completion graph, intuitively it must be shown that we can determine the propagation of labels due to the addition in all complete and clash-free completion graphs for a KB by simply using the summary completion graph. The main idea of the approach is that given a set of assertions β , the structures for β can be added to the summary completion graph and the expansion rules can be applied to the added labels in a similar manner as when incrementally updating a complete graph (as in Section 4). Then, it can be shown that the propagation of labels to root nodes in the summary completion graph subsumes the propagation in all completion graphs. Further, we show that the concept guide paths must also exist in the summary completion graph; collectively, this implies the completeness of the approach.

Due to the specialized treatment of the \sqcup -rule, there may be labels present in the summary completion graph which prohibit the application of an expansion rule. For example, β may include a type assertion of the form $\forall R.C(a)$ and in the summary completion graph all R -neighbors of a already have C in their label; however, in the different complete and clash-free completion graphs for the KB prior to the addition of β , there could exist some R -neighbor that does not contain C in its label (this is due to the non-determinism in the tableau algorithm). In this case, this neighbor would in fact have a label change, causing it to be considered when detecting concept guide paths. This problem is overcome by a modification when checking if the expansion rules can be applied to a node. Specifically, if when updating the summary completion graph with β , it is the case that a node label exists which prevents the application of an expansion rule, then it is applied anyway. In order to ensure that the algorithm still terminates, rule applications are tracked using a marking function θ when they have been applied to a specific node during

the update of the summary completion graph (shown in Table 2). Therefore, the re-application will only happen once. Given this, the approach for updating of the summary completion graph is defined as follows.

| | |
|--------------------|--|
| \sqcap -rule: | if 1) $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked and 2) either a) $\theta((C_1 \sqcap C_2, x)) == false$ or b) $\{C_1, C_2\} \notin \mathcal{L}(x)$ then set $\theta((C_1 \sqcap C_2, x)) = true$ and $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$ |
| \sqcup -rule: | if 1) $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked and 2) either a) $\theta((C_1 \sqcup C_2, x)) == false$ or b) $\{C_1, C_2\} \notin \mathcal{L}(x)$ then set $\theta((C_1 \sqcup C_2, x)) = true$ and $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$ |
| \exists -rule: | if 1) $\exists S.C \in \mathcal{L}(x)$, x is not blocked and 2) either a) $\theta((\exists S.C, x)) == false$ or b) x has no S -neighbor y with $C \in \mathcal{L}(y)$ then set $\theta((\exists S.C, x)) = true$ and create a new node y with $\mathcal{L}(\langle x, y \rangle) = S$ and $\mathcal{L}(y) = C$ |
| \forall -rule: | if 1) $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked and 2) either a) $\theta((\forall S.C, x, y)) == false$ and there is an S -neighbor y of x with $C \in \mathcal{L}(y)$ or b) there is an S -neighbor y of x with $C \notin \mathcal{L}(y)$ then set $\theta((\forall S.C, x, y)) = true$ and $\mathcal{L}(y) = \mathcal{L}(y) \cup C$ |
| \forall_+ -rule: | if 1) $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked and 2) there is some R with $\text{Trans}(R)$ and $R \boxplus S$, 3) either a) $\theta((\forall S.C, R, x, y)) == false$ and there is an R -neighbor y of x with $\forall R.C \in \mathcal{L}(y)$ b) there is an R -neighbor y of x with $\forall R.C \notin \mathcal{L}(y)$ then set $\theta((\forall S.C, R, x, y)) = true$ and $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$ |

Table 2

Modified Tableau Expansion Rules for the Summary Completion Graph.

Definition 16 (*Summary Completion Graph Update*): Let \mathbf{K} be a *SHI KB*, S_G be the summary completion graph for \mathbf{K} , and β a set of *ABox assertions*. Then S_G is incrementally updated with β using the approach discussed in Section 4, however:

- *clashes are ignored*
- *the modified expansion rules defined in Table 2 are assumed and are applied to the following nodes:*
 - (1) *each node x_a corresponding to some individual $a \in \mathbf{I}_\beta$*
 - (2) *any node subsequently reached by the application of an expansion rule due to condition 1–3*
 - (3) *any node that was previously blocked, yet the block is invalidated because of the addition of a node label due to condition 1–3*

Termination of the update process is shown in Appendix A.2.5 of [26]. Denote by $update(\beta, S_G)$ the update of summary completion graph S_G with β according to Definition 16. Additionally, denote by $Dep(\beta, S_G)$ the set of named individuals whose corresponding root nodes have a node label, or incoming/outgoing edge or edge label that is (re)added during $update(\beta, S_G)$. In order to show completeness of the approach, we must demonstrate that S_G can be used to find all $a \in Dep(\beta, G, G')$ for some $G \in \text{Comp}(\mathbf{K})$ and G' the result of adding β to G . First, note that if an edge $\langle x, y \rangle \in \mathcal{E}$, where x, y are root nodes, is added to G , it must have been added due to

a role assertion in β (the same holds for edge labels for edges between root nodes); this is due to the fact that the tableau expansion rules do not add edges or edge labels between named individuals. Therefore, it suffices to show that if a label is added to some root node x_a during the update of some $G \in \text{Comp}(\mathbf{K})$, then a label will be (re)added to the corresponding root node in the summary completion graph.

Lemma 1 *Let \mathbf{K} be a SHI KB, $G \in \text{Comp}(\mathbf{K})$, S_G be the summary completion graph for \mathbf{K} , and β a set of ABox assertions. If when adding β to G , a root node x has a concept name added to $\mathcal{L}(x)$, then x will have a concept name (re)added to $\mathcal{L}_{S_G}(x)$ when updating S_G with β .*

The proof for this lemma can be found in Appendix A.2.6 of [26]. Given this, the summary completion graph can be used to determine $\text{Dep}(\beta, G, G')$ by tracking the nodes that are reached during the update of the summary completion graph. However, one must then find the concept guide paths involving these individuals in each G' (i.e., the result of adding β to $G \in \text{Comp}(\mathbf{K})$). Once again, it can be shown that it suffices to simply use the updated summary completion graph, rather than all G' . This is implied by the following theorem; again, the proof for this lemma is omitted, however it can be found in Appendix A.2.7 of [26].

Lemma 2 *Let \mathbf{K} be a SHI KB, $G \in \text{Comp}(\mathbf{K})$, S_G be the summary completion graph for \mathbf{K} , C be a SHI concept that is safe with respect to \mathbf{K} , and $\mathcal{G} = \text{guide}(\neg C)$. If for some $a \in \mathbf{I}_{\mathbf{K}}$, $n, m \in N_{\mathcal{G}}$ s.t. $\neg C \in \mathcal{L}_{\mathcal{G}}(n)$ there is a concept guide path $\text{path}(n, m, x_a, x_b, \mathcal{G}, G)$, then there is a concept guide path $\text{path}(n, m, x_a, x_b, \mathcal{G}, S_G)$.*

Now let us consider the third condition of Definition 14 for addition updates. An approach must be developed that uses the summary completion graph to determine the clashes observed when adding β to $G \in \text{Comp}(\mathbf{K})$ that are dependent on some $D_1 \sqcup D_2 \in \mathcal{L}(x)$; additionally, the root nodes with a label that is also dependent on $D_1 \sqcup D_2 \in \mathcal{L}(x)$ in a completion graph $G' \in \text{Comp}(\mathbf{K}) \setminus G$ must also be determined. First, it can be shown that all clashes observed when updating each $G \in \text{Comp}(\mathbf{K})$ will be observed on some node when updating the summary completion graph. While clashes are ignored during the construction and update of the summary completion graph, they will be present and can be tracked. This is not formally shown here, however it is a direct consequence of Lemma 7 in [26].

Next, we must show that the disjunctions that clashes are dependent on can be located using the summary completion graph. Further, we must show that we can locate the root nodes with a node label dependent on these disjunctions. The general idea of the approach, and the focus of the remainder of this section, is that a specialized disjunction dependency function can be introduced for the summary completion graph, such that the dependencies subsume those in all completion graphs. Due to the fact that labels from different completion graphs are essentially merged in the summary completion graph, care must be taken when maintaining the dependence of labels on disjunctions. In particular, we must ensure that if a label *could*

be added due to a disjunction, then this is reflected in the disjunction dependencies. To account for this, a disjunction dependency function for the summary completion graph is defined in Definition 17.

Definition 17 (*Summary Disjunction Dependency*) Given summary completion graph S_G and node x with $D_1 \sqcup D_2 \in \mathcal{L}(x)$, inductively define the set S of concept/node pairs (C, x) that are dependent on $D_1 \sqcup D_2 \in \mathcal{L}(x)$ as follows:

- $(D_1 \sqcup D_2, x) \in S$
- if $(C, y) \in S$, then if C of the form:
 - (1) $C_1 \sqcap C_2$, then $\{(C_1, y), (C_2, y)\} \subseteq S$
 - (2) $C_1 \sqcup C_2$, then $\{(C_1, y), (C_2, y)\} \subseteq S$
 - (3) $\exists R.D$, then for each z s.t.
 - (a) z a R -neighbor of y and $D \in \mathcal{L}(z)$, then $(B, z) \in S$ for all $B \in \mathcal{L}(z)$
 - (b) z a R -neighbor of y , z blocked by m and $D \in \mathcal{L}(m)$, then $(B, m) \in S$ for all $B \in \mathcal{L}(m)$
 - (c) z a R -neighbor of y , z blocked by m , $\forall S.D \in \mathcal{L}(m)$ s.t. $\text{Inv}(R) \sqsubseteq S$, and $D \in \mathcal{L}(m)$, then $(B, y) \in S$ for all $B \in \mathcal{L}(y)$
 - (4) $\forall R.D$, then for each z s.t.
 - (a) z a R -neighbor of y and $D \in \mathcal{L}(z)$, then $(D, z) \in S$
 - (b) z a R -neighbor of y , z blocked by m and $D \in \mathcal{L}(m)$, then $(D, m) \in S$
 - (c) y blocks z , m the predecessor of z , m a R -neighbor of z , and $D \in \mathcal{L}(m)$, then $(D, m) \in S$
 - (5) $\forall R.D$ and there is some P s.t. $\text{Trans}(P)$ and $P \sqsubseteq R$, then for each z s.t.
 - (a) z a P -neighbor of y and $\forall P.D \in \mathcal{L}(z)$, then $(\forall P.D, z) \in S$
 - (b) z a P -neighbor of y , z blocked by m and $\forall P.D \in \mathcal{L}(m)$, then $(\forall P.D, m) \in S$
 - (c) y blocks z , m the predecessor of z , m a P -neighbor of z , and $\forall P.D \in \mathcal{L}(m)$, then $(\forall P.D, m) \in S$

Denote by $\text{Disj}_{S_G}(D_1 \sqcup D_2, x)$ the named individuals whose corresponding root nodes in S_G have a node label dependent on $D_1 \sqcup D_2 \in \mathcal{L}(x)$. The disjunction dependency function can easily be constructed after building or updating a summary completion graph. Further, if a new disjunction is introduced after an update, then its dependencies can be easily identified after the update. Given this, it is simply assumed that the disjunction dependency function is maintained.

Importantly, it can be shown that if a clash is observed when updating some $G \in \text{Comp}(\mathcal{K})$ that is dependent on some disjunction D , then the clashes observed when updating the summary completion graph will be dependent on disjunctions whose dependencies subsume the root nodes that are dependent on D in some completion graph. This is not formally shown here, as it is a direct consequence of Lemmas 2, 3, & 6 of [26]. However, this implies that we can find all individuals a s.t. there was a clash is observed when updating G that is dependent on $D_1 \sqcup D_2 \in \mathcal{L}(y)$ and $a \in \text{Disj}_{\mathcal{K}}(D_1 \sqcup D_2, y)$. Given this, to take into account the third condition of Definition 14, we must then find concept guide paths involving these individuals (or some

individual $Dep(\beta, G, G')$ in some $G' \in Comp(\mathbf{K}) \setminus G$; it is a direct consequence of Lemma 2 that this can be performed simply using the summary completion graph.

ABox Deletions. Until this point, we have only discussed how the summary completion graph can be used to support addition updates. Given the axiom tracing function discussed in Section 4, incremental deletions can be supported in a similar manner. This is because the completeness of the axiom tracing function when applying it to the summary completion graph directly follows from Theorem 1 of [26]. Given this, we can revert the change events in the summary completion graph that are dependent on the deletion, and then the necessary expansion rules can be applied to the summary completion graph; importantly, the modified \sqcup -rule must be used (as presented in Definition 15) and clashes must be ignored during the re-application of the expansion rules. Thus, the summary completion graph for $\mathbf{K} - \beta$ is easily obtained; for ease of presentation denote this process by $Del(\beta, S_G)$. Therefore, deletion updates can be supported by then adding β back to the summary completion corresponding to $\mathbf{K} - \beta$ and performing the same approach as in the case for additions. It is noted that after a deletion has been processed using this technique, the summary completion graph must again be updated to reflect the deletion. As in the original retraction of β , the axiom tracing function can be used for this purpose. Given this, an over-estimate of the concept candidates is introduced.

Definition 18 (*Concept Candidates Overestimate*): Let \mathbf{K} be a *SHI* KB, β an *ABox* addition (or deletion), S_G the summary completion graph for \mathbf{K} , $S'_G = Del(\beta, S_G)$, $S''_G = update(\beta, S_G)$ (respectively $S''_G = update(\beta, S'_G)$), C a *SHI* concept that is safe with respect to \mathbf{K} and β , and $\mathcal{G} = guide(\neg C)$. Define the overestimate of candidate individuals, denoted $CC_{S_G}(\mathbf{K}, C, \beta)$, to be the set of named individuals $a \in \mathbf{I}_{\mathbf{K}} \cup \mathbf{I}_{\beta}$ such that for some $n, m \in N_{\mathcal{G}}$ s.t. $\neg C \in \mathcal{L}_{\mathcal{G}}(n)$, one of the following conditions is satisfied:

- (1) $a \in \mathbf{I}_{\beta}$
- (2) $b \in Dep(\beta, S_G)$ (respectively $b \in Dep(\beta, S'_G)$) and there is a concept guide path $path(n, m, x_a, x_b, \mathcal{G}, S''_G)$
- (3) the expansion rules are applied to a node x during $update(\beta, S_G)$ (respectively $update(\beta, S'_G)$) such that $\{A, \neg A\} \subseteq \mathcal{L}(x)$, $A \in \mathcal{L}(x)$ or $\neg A \in \mathcal{L}(x)$ is dependent on $D_1 \sqcup D_2 \in \mathcal{L}(y)$ (determined using Definition 17), and for some $b \in Dis_{S''_G}(D_1 \sqcup D_2, y)$ there is a concept guide path $path(n, m, x_a, x_b, \mathcal{G}, S''_G)$

Lastly, it can be shown that the over-estimate is complete; again, the proof for this theorem is omitted, however it can be found in Appendix A.2.10 of [26].

Theorem 4 Given a *SHI* KB \mathbf{K} , *ABox* update β , *SHI* concept C that is safe with respect to \mathbf{K} and β , and summary completion graph S_G for \mathbf{K} , then $CC(\mathbf{K}, C, \beta) \subseteq CC_{S_G}(\mathbf{K}, C, \beta)$.

5.3 Supporting Complex Query Patterns

Thus far, the techniques presented have only addressed queries that are simply composed of a single DL concept. Given this, the approach is now extended to support complex query patterns. As discussed previously, the general approach for supporting complex query patterns is to transform each role atom in the query into a concept atom, which is referred to as *rolling-up* the query. It can be shown that the techniques developed in the previous sections can be used to find the candidate bindings for the distinguished variable that the query is rolled-up into if the resulting query concept is safe w.r.t. the KB and updates. Further, it can be shown that the query can simply be rolled-up once using a single set of new concept names and the additional type assertions for the representative concepts can be ignored when determining the candidates. Thus, if the query can be rolled-up into a distinguished variable x , the same approach can be used to find the candidates using the single rolled-up concept.

Given a retrieval query Q with $DVar(Q) = \{x_1, \dots, x_n\}$, denote the rolling-up of Q into $x_i \in DVar(Q)$ by $Rollup(i, Q)$, such that it produces a *SHI* concept C where each x_j , $i \neq j$, has been replaced by a new atomic concept D_j not appearing in the KB, any update, or the query. It is assumed that the concept obtained by $Rollup(i, Q)$ is safe with respect to K and all updates. Given this, for a new (respectively invalidated) binding $\{a_1, \dots, a_n\}$ to occur, it must be the case that the individual bound to the distinguished variable that the query is rolled-up into is in the set of concept candidates for the rolled-up query concept.

Theorem 5 *Let K be a SHI KB, Q a conjunctive retrieval query that can be rolled-up into a distinguished variable $x_i \in DVar(Q)$, $C = Rollup(i, Q)$, and β an ABox addition (or deletion). If $K \not\models Q[x_1/a_1, \dots, x_n/a_n]$ and $K+\beta \models Q[x_1/a_1, \dots, x_n/a_n]$ (respectively $K \models Q[x_1/a_1, \dots, x_n/a_n]$ and $K - \beta \not\models Q[x_1/a_1, \dots, x_n/a_n]$), then $a_i \in CC(K, C, \beta)$.*

Again, the proof for this theorem is omitted, however it can be found in Appendix A.2.11. It is a direct consequence of Lemma 2, Theorems 4 & 5 and the fact that each D_j is a new atomic concept that the summary completion graph can be used to determine an overestimate of concept candidates for the rolled-up query concept.

It is easy to show, however, that one cannot simply consider the candidates for the variable that the query is rolled-up into as the only candidates for the other distinguished variables in the query. This is because the previously described techniques do not allow us to make any statements regarding the candidates for any variables in the query except for the distinguished variable that it is rolled-up into. Therefore, we develop a technique determine the remaining candidates; we refer to this as the *query impact* on the candidates. It is first pointed out that given Theorem 5 and the monotonicity of *SHI*, in the event of deletions, all that must be considered after

the update are the previous answer sets which have some individuals that is in the set of query concept candidates. Therefore, one simply needs to re-check these answer sets to ensure that the entailment still holds. Given this, the remainder of this section only addresses ABox additions.

It has previously been shown that a conjunctive query can be answered by syntactically *mapping* the query into all completion graphs for the KB [38]. More specifically, [38] defines a syntactic mapping from a query Q (restricted to only simple roles) into a completion graph G , denoted $Q \hookrightarrow G$, using a mapping μ from the variables (both distinguished and non-distinguished) and individuals in Q into the nodes of G such that:

- $\mu(a) = x_a$ for each individual $a \in Q$,
- for each atom $C(x)$ in Q , $C \in \mathcal{L}(\mu(x))$, and
- for each atom $R(x,y)$ in Q , $\mu(y)$ is an R -neighbor of $\mu(x)$

If the query can be mapped into all completion graphs, then the KB satisfies the query [38]. In order for such an approach to be complete, a special blocking condition, tree-blocking, must be used during the tableau algorithm, in which blocking is delayed to take into account the longest path in the query; this ensures that such a mapping will be possible in the presence of blocking. It is important to note that if the query contains only distinguished variables, then tree-blocking is not necessary and simply dynamic blocking can be used. This is because root nodes corresponding to named individuals are never blocked during the tableau algorithm. Additionally, a TBox axiom $\top \sqsubseteq C \sqcup \neg C$ must be added to the KB for each concept atom C in the query; this is necessary as the query concepts are syntactically mapped into the completion graph.

A straightforward application of this technique can be leveraged for our purpose. First, it is noted that extending the KB with $\top \sqsubseteq C \sqcup \neg C$ for each query concept may be impractical in the syndication framework when dealing with a substantial number of registered subscriptions. Therefore, a slight modification of the approach is used, in which the mapping of concept names in the query is ignored. Specifically, given a conjunctive retrieval query Q with $DVar = \{z_1, \dots, z_n\}$ that has been rolled-up into a distinguished variable $z_i \in DVar(Q)$ resulting in concept C and the set of query concept candidates $CC(K, C, \beta)$, the following mapping is checked in some $G \in Comp(K + \beta)$ for each x_b s.t. $b \in CC(K, C, \beta)$:

- $\mu(a) = x_a$ for each individual $a \in Q$,
- $\mu(z_i) = x_b$,
- $\mu(z_j) = x_c$ for $1 \leq j < i, i < j \leq n$ and some root node x_c
- for each atom $R(x,y)$ in Q , $\mu(y)$ is an R -neighbor of $\mu(x)$

If the original candidate individual b cannot be mapped into x_b such that there is a valid mapping for the remaining query nodes, then this individual does not need to be considered as a candidate. This follows as a completion graph (i.e., model)

has just been found in which the query cannot be mapped [38]. However, if the query can be mapped into the completion graph such that the candidate individual, b , is mapped into x_b , then this individual must be considered as a candidate binding. Importantly, any named individual in the completion graph that can be mapped into the remaining distinguished variables of the query graph must also be added to the candidate set. The mapping only needs to be performed for one completion graph, as for the entailment to occur, it must be satisfied by all completion graphs. Given a set of individuals A and KB K , denote by $map_impact(A, Q, \beta)$ the set of all named individuals corresponding to the root nodes that are mapped into a distinguished variable in a valid mapping.

To illustrate the approach, consider the query $(x, y) \leftarrow Company(x) \wedge onSellList(x, y) \wedge hasCEO(y, z)$. Let us assume that the query is rolled up into the variable x ; then, when determining the additional candidates, any individual $a \in CC(K, C, \beta)$ that does not have a *onSellList*-neighbor, which in turn has a *hasCEO* neighbor cannot be a candidate binding for the variable x . Again, this is because a model has just been found in which the entailment does not hold. However, if a has *onSellList*-neighbors b and c (both of which are root-nodes) that also have *hasCEO*-neighbors d, e respectively, then a, b, c are considered in the candidate set. Lastly, it can be shown that the query impact approach is complete; the proof for this theorem can be found in Appendix A.2.12 of [26].

Theorem 6 *Let K be a SHI KB, Q a conjunctive retrieval query that can be rolled-up into a distinguished variable $x_i \in DVar(Q)$, $C = Rollup(i, Q)$, β an ABox addition, and $A = CC(A, Q, \beta)$. If $K \not\models Q[x_1/a_1, \dots, x_n/a_n]$ and $K + \beta \models Q[x_1/a_1, \dots, x_n/a_n]$, then $\{a_i, \dots, a_n\} \subseteq map_impact(A, Q, \beta)$.*

5.4 Finding Concept Guide Paths

Until now, the task of finding concept guide paths in a completion graph has not been addressed. It is a fairly straightforward observation that the task of determining if there is a concept guide path in the summary completion graph can be reduced to evaluating regular path expressions over a labeled directed graph. This follows as the summary completion graph is a labeled directed graph, and it is easily seen that a set of regular path expressions can be constructed from the concept guide; specifically, for each node x in the concept guide, a regular path expression can be created that starts at the node that corresponds to the negated query concept and terminates at x . Then, in the most naïve approach, for each pair of nodes in the completion graph, one needs to test if there is a path in the completion graph that satisfies one of the regular path expressions.

There exist known results regarding the complexity of evaluating regular path expression over graph databases. Specifically, it has been shown that deciding if a

graph G contains a directed path from nodes x to y satisfying regular expression R can be performed in polynomial time [35]. Given this, assume there are m nodes in the concept guide and s nodes in the summary completion graph. Then, there are $\binom{s}{2}$ pairs of nodes in the summary completion graph, which is $O(n^2)$. Thus, the concept guide paths can be found in $O(m * n^2)$ time. From a practical point of view, it is important to note that there has been recent work in XML database literature that addresses the evaluation of regular path expressions over graph-based XML data (XML documents with IDREFs) [30]. Therefore, there exist known algorithms which can be utilized to locate concept guide paths.

5.5 Incremental Query Answering Algorithm

Prior to presenting the incremental query answering algorithm, the pseudo code for updating the summary completion graph and determining the set of individuals that must be considered in concept guide paths when determining the concept candidates is provided (shown in Algorithm 1). It is assumed that the summary completion graph S_G is created at startup. For ease of exposition, given $S'_G = \text{update}(\beta, S_G)$ and a clash c observed during $\text{update}(\beta, S_G)$ that is dependent on a set of disjunctions in node labels of S'_G , denote by $\text{Disj}(c)$ the set of named individuals a s.t. $a \in \text{Disj}_{S'_G}(D_1 \sqcup D_2, x)$ for some disjunction $D_1 \sqcup D_2 \in \mathcal{L}_{S'_G}(x)$ that c is dependent on.

Algorithm 1 $\text{update_summary}(S_G, \beta)$

Input:

S_G : Summary completion graph
 β : Set of ABox assertions

Output:

AI : Set of named individuals
 S_G : Updated summary completion graph

```

1: if  $\beta$  is a deletion then
2:    $S_G \leftarrow \text{Del}(\beta, S_G)$ 
3: end if
4:  $AI \leftarrow \mathbf{I}_\beta$ 
5:  $S_G \leftarrow \text{update}(\beta, S_G)$ 
6: for all root nodes  $x_a$  such that the tableau expansion rules are applied to  $x_a$  during
    $\text{update}(\beta, S_G)$  do
7:    $AI \leftarrow AI \cup \{a\}$ 
8: end for
9: for all clashes  $c$  observed during  $\text{update}(\beta, S_G)$  do
10:   $AI \leftarrow AI \cup \text{Disj}(c)$ 
11: end for
12: return  $AI, S_G$ 

```

The general algorithm for incremental query answering is presented in Algorithm 2. The algorithm is presented in terms of a single query. It is assumed the query can

be rolled up into the distinguished variable $x_i \in DVar(Q)$ and that the initial set of answers for Q_c is previously determined. The algorithm first integrates the update into a consistent KB; if the update is an addition, it is assumed that KB is consistent after the update. Following this, the set of candidate individuals is found using the summary completion graph and concept guide searches (lines 2–3); for simplicity, given a set of individuals AI and concept guide \mathcal{G} , the location of concept guide paths is denoted as $guide_search(AI, \mathcal{G})$. If the update is an addition, the remaining candidates are found by taking into account the query impact. After this, the set of candidate distinguished variable bindings is iterated over and checked for entailment. Standard techniques for query answering are used. In contrast, if the update is a deletion, each tuple in the previous answer set is iterated over and tuples that do not contain some individual in the set of affected individuals are still entailed, as the conditions for the invalidation of the entailment were not satisfied; otherwise, the tuples are re-checked for entailment. Lastly, correctness of Algorithm 2 can be shown. The proof for this theorem is omitted, however it can be found in Appendix A.2.13 of [26].

Theorem 7 *Algorithm 2 is sound, complete, and terminating.*

6 Empirical Results

A prototype of the algorithm developed in this paper has been implemented as an extension to the OWL DL reasoner Pellet¹³. In order to evaluate the algorithm for the purpose of OWL-based syndication, an empirical evaluation has been performed using various OWL KBs with large ABoxes; namely VICODI¹⁴, SEMINTEC¹⁵, the Lehigh University Benchmark (LUBM)¹⁶, and an extension to LUBM, called the University Ontology Benchmark (UOB)¹⁷ have been used. These ontologies have been selected as a test suite because they are expressed in the DLs that the algorithm supports and provide a range of expressivity¹⁸. Also, the constructs used in these ontologies align with common usage of OWL constructs [47]. It is noted that the ABoxes have been manually created¹⁹ or are automatically generated using dataset generators accompanying the ontologies. This allows the simulation of

¹³ Pellet Project Homepage: <http://pellet.owldl.com/>

¹⁴ VICODI project homepage: <http://www.vicodi.org/>

¹⁵ SEMINTEC project homepage: <http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm>

¹⁶ LUBM project homepage: <http://swat.cse.lehigh.edu/projects/lubm/>

¹⁷ Developed by IBM and is available through their Integrated Ontology Development Toolkit: <http://www.alphaworks.ibm.com/tech/semanticstk>

¹⁸ Because the approach is applicable to *SHI*, functional role assertions have been removed from the SEMINTEC and UOB ontologies.

¹⁹ We would like to acknowledge Boris Motik for his creation of the larger VICODI and SEMITEC datasets (described in [36]).

Algorithm 2 *update_query_results*($K, S_G, Q, \mathcal{G}, R, \beta$)

Input:

K : Consistent *SHI* KB
 S_G : Summary completion graph for K
 Q : Conjunctive query
 \mathcal{G} : Concept guide for $\text{Rollup}(i, Q)$
 R : Set of all current bindings (answer set)
 β : ABox update

Output:

K : Consistent *SHI* KB
 S_G : Updated summary completion graph
 R : Updated bindings (answer set)

```
1:  $K \leftarrow K \oplus \beta$ 
2:  $AI, S_G \leftarrow \text{update}(S_G, \beta)$ 
3:  $CC \leftarrow \text{guide\_search}(AI, \mathcal{G})$ 
4: if  $\beta$  is an addition then
5:    $QC \leftarrow \text{map\_impact}(CC, Q, \beta)$ 
6:   for all  $\{a_1, \dots, a_n\} \in QC^n$  s.t.  $\#DVar(Q) = n$  do
7:     if  $K \models Q_c[x_1/a_1, \dots, x_n/a_n]$  then
8:        $R \leftarrow R \cup \{(a_1, \dots, a_n)\}$ 
9:     end if
10:  end for
11: else if  $\beta$  is an deletion then
12:  for all  $(a_1, \dots, a_n) \in R$  do
13:    if  $CC \cap \{a_1, \dots, a_n\} = \emptyset$  then
14:      continue
15:    else if  $K \not\models Q_c[x_1/a_1, \dots, x_n/a_n]$  then
16:       $R \leftarrow R \setminus \{(a_1, \dots, a_n)\}$ 
17:    end if
18:  end for
19:   $S_G \leftarrow \text{Del}(\beta, S_G)$ 
20: end if
21: return  $K, S_G, R$ 
```

background information and publications which persist in the broker’s KB for a long period of time, and allows us to investigate the way in which the incremental query answering algorithm scales. Table 3 presents an overview of the ontologies, including DL expressivity, number of classes, properties, individuals, and triples (we will comment on the last two columns shortly).

| KB | Exp. | # Clas. | # Prop. | # Ind. (KB1 / KB2) | # Triples (KB1 / KB2) | Sum. Build (sec) (KB1 / KB2) | Sum. Mem. (mb) (KB1 / KB2) |
|------|--------------|---------|---------|-----------------------|--------------------------|---------------------------------|-------------------------------|
| VIC | <i>ALHI</i> | 194 | 29 | 16,942 / 33,884 | 54,081 / 107,734 | 1.7 / 3.7 | 57 / 115 |
| SEM | <i>ALCIF</i> | 59 | 16 | 17,941 / 35,882 | 65,560 / 130,784 | 2.4 / 5.3 | 82 / 165 |
| LUBM | <i>SHI</i> | 43 | 63 | 16,283 / 37,450 | 130,800 / 225,095 | 2.3 / 9.2 | 59 / 228 |
| UOB | <i>SHIF</i> | 51 | 77 | 25,272 / 51,762 | 246,266 / 423,031 | 15.2 / 30.4 | 183 / 331 |

Table 3
Test-Suite Ontology Overview.

For each of the ontologies, a set of queries has been used to simulate subscriptions. For the LUBM and UOB benchmarks, a set of sample queries accompanies the benchmarks and a representative subset of these queries are used. In contrast, the VICODI and SEMINTEC ontologies do not have such a query suite, however, there has been recent work in literature [36] in which a set of queries has been obtained from the authors of these ontologies. Therefore, these queries have been used, as they will provide insights into response times for expected queries over these datasets. In the experiments, varying sized ABox additions and deletions were randomly selected from each dataset. Update sizes include 1, 5, 15, 25, and 50 assertions. These sizes were selected as they align with publication sizes expected in realistic syndication systems. In the experiments, the initial set of answers for the query was first determined, and then the randomly selected assertions were added (or removed) to the KB and the query results were updated. The aim behind the evaluation was to simulate new publications arriving at the syndication broker.

Two versions of the OWL DL reasoner Pellet have been used in the evaluation; a regular version of the reasoner and a version that has been extended with the algorithm to reduce the candidate individuals. The DL reasoner RacerPro²⁰ was also used in the evaluation. Similar to Pellet, RacerPro is a highly optimized tableau-based DL reasoner, however RacerPro is sound and complete for the DL *SHIQ*. Lastly, the KAON2²¹ OWL reasoner was also used in the evaluation; similar to RacerPro, KAON2 supports reasoning for the DL *SHIQ*. Interestingly, KAON2 is not a tableau-based reasoner, but rather it reduces OWL KBs to disjunctive datalog and is highly optimized for ABox reasoning [36]. Our aim in using KAON2 in evaluation was to gain insights into tableau-based algorithms for syndication purposes when compared to other possible approaches.

²⁰ RacerPro is commercially supported by Racer Systems GmbH & Co. KG: <http://www.racer-systems.com/index.phtml>

²¹ KAON2 project homepage: <http://kaon2.semanticweb.org/>

The experiments were run on a Linux machine with 2GB of RAM and a 3.06GHz Intel Xeon CPU. Pellet v1.5, RacerPro v1.9.0, and KAON2 release 2007-09-07 were used and all results were averaged over 75 iterations. Note that in all of the figures showing the query response time results, the X-axis corresponds to the update size, and the Y-axis is the response time in milliseconds for query answering (the scale is logarithmic). Additionally, the response times shown only include the time to determine the query results; the response times do not include consistency checking times or query preparation times performed by Pellet (this is not utilized in the incremental version of Pellet) and RacerPro. The response times for the incremental query algorithm developed in this paper include both the time to find the candidates and execute the query over the candidate set (i.e., Algorithm 2). In the experiments a maximum response time of 100 seconds was imposed in the tests, as any time greater than this will clearly not scale for high-demand syndication purposes.

As discussed earlier, the summary completion graph must be built so that the technique can be used. The total time to construct the initial summary completion graph is of interest, as well as the potential memory overhead imposed by the structure. Table 3 shows the total time (in seconds) to construct the initial summary completion for each of the different datasets, as well as the memory overhead. As expected, the initial construction of the summary completion graphs introduces overhead. However, the process must only be performed once at startup. It is also clear that there is a memory impact of the summary completion graph. However, in the implementation of the algorithm, the overlap of between the summary completion graph and the cached completion graph corresponding to a model of the KB is not exploited. Today’s tableau-based reasoners typically cache the completion graph constructed during the initial consistency check, as it is used in optimizations for other reasoning tasks (e.g., classification). A substantial portion of the structure in the cached completion graph will overlap with the summary completion graph. Therefore, with further engineering, this memory overhead can be potentially reduced.

For the VOCODI ontology, the following queries have been used:

- (1) $(x) \leftarrow \textit{Individual}(x)$
- (2) $(x, y, z) \leftarrow \textit{Military} - \textit{Person}(x) \wedge \textit{hasRole}(y, x) \wedge \textit{related}(x, z)$
- (3) $(x, y) \leftarrow \textit{Military} - \textit{Person}(x) \wedge \textit{hasRole}(y, x)$

Queries 1 and 2 have been used in previous literature [36] to evaluate DL query answering and were suggested by the ontology authors. Query 3 has been included as it provides additional insights into the approach (discussed later). The response times for the queries are shown in Figure 2 (note that *Inc-Pellet* corresponds to Algorithm 2). Let us consider query 1; the query answering times for the regular version of Pellet is between 1.4 to 3.08 seconds, depending on the dataset size. The response time remains comparable through update sizes, as the query is re-performed

from scratch and the update sizes are small relative to the overall KB size. Response time for RacerPro exhibits similar properties as Pellet. The incremental query answering approach demonstrates substantial performance improvements over both the regular version of Pellet and RacerPro. For both datasets, approximately 1.5 to 3 orders of magnitude performance improvements over Pellet are exhibited, and the response time is in the 10s of milliseconds (in many cases less than 10ms). This is clearly due to the reduction in the portion of the KB that must be considered for the query after the update. Table 4 presents the actual number of candidates for addition updates using the approach for the different update sizes and queries (the results for deletions are very similar and are therefore omitted). This table also shows the percentage of the KB that this candidate set represents (shown in parenthesis). For the first query, the technique provides a dramatic reduction in the portion of the KB that must be considered (always below 1% of the original KB). It can also be seen in Figure 2 that as the update size is increased, the performance of the approach scales well. Deletions take slightly longer than additions because the deleted assertions must be retracted from the summary completion graph, whereas this process is not necessary for additions. KAON2 outperforms the regular tableau-based reasoners (i.e., Pellet and RacerPro) and exhibits comparable results through the update sizes. However, the incremental version of Pellet performs better than KAON2 in this experiment.

Figure 2 also presents the results for queries 2 and 3. In the second query, the query answering times for the regular version of Pellet and RacerPro exhibit similar characteristic as for the first query. Further, KAON2 again exhibits comparable response times through updates sizes and in general performs better than in query 1. However, the incremental approach does not provide as dramatic performance improvements as it did in the first query. This can be explained by inspecting Table 4; in particular, it can be seen that a large portion of the knowledge base is considered after each update, and therefore, it is like re-running the query from scratch.

If we inspect the results of query 3, additional insights into the results for the second query are provided. Query 3 is actually a subset of the second query, in which the last role atom from query 2 is excluded. Once again, the incremental algorithm exhibits dramatic performance improvements, as the candidates considered are a small subset of the original KB. This sheds light on the previous query, as there are individuals in the KB that are related to an extremely large portion of the KB by the *related* role. Therefore, when taking into account the query impact, almost all of the knowledge base is included as a candidate. As we will see in the remainder of the evaluation, this was the only query for any of the ontologies that demonstrated this behavior, indicating the approach should be effective in general. Lastly, it is noted that the average incremental consistency checking times for this ontology can be found in Chapter 5 of [26]. A detailed presentation of these results is omitted here, however on average it was always below 10 milliseconds for addition updates and just over 10 milliseconds for deletions. Clearly, this indicates the practicality of the approach for syndication purposes.

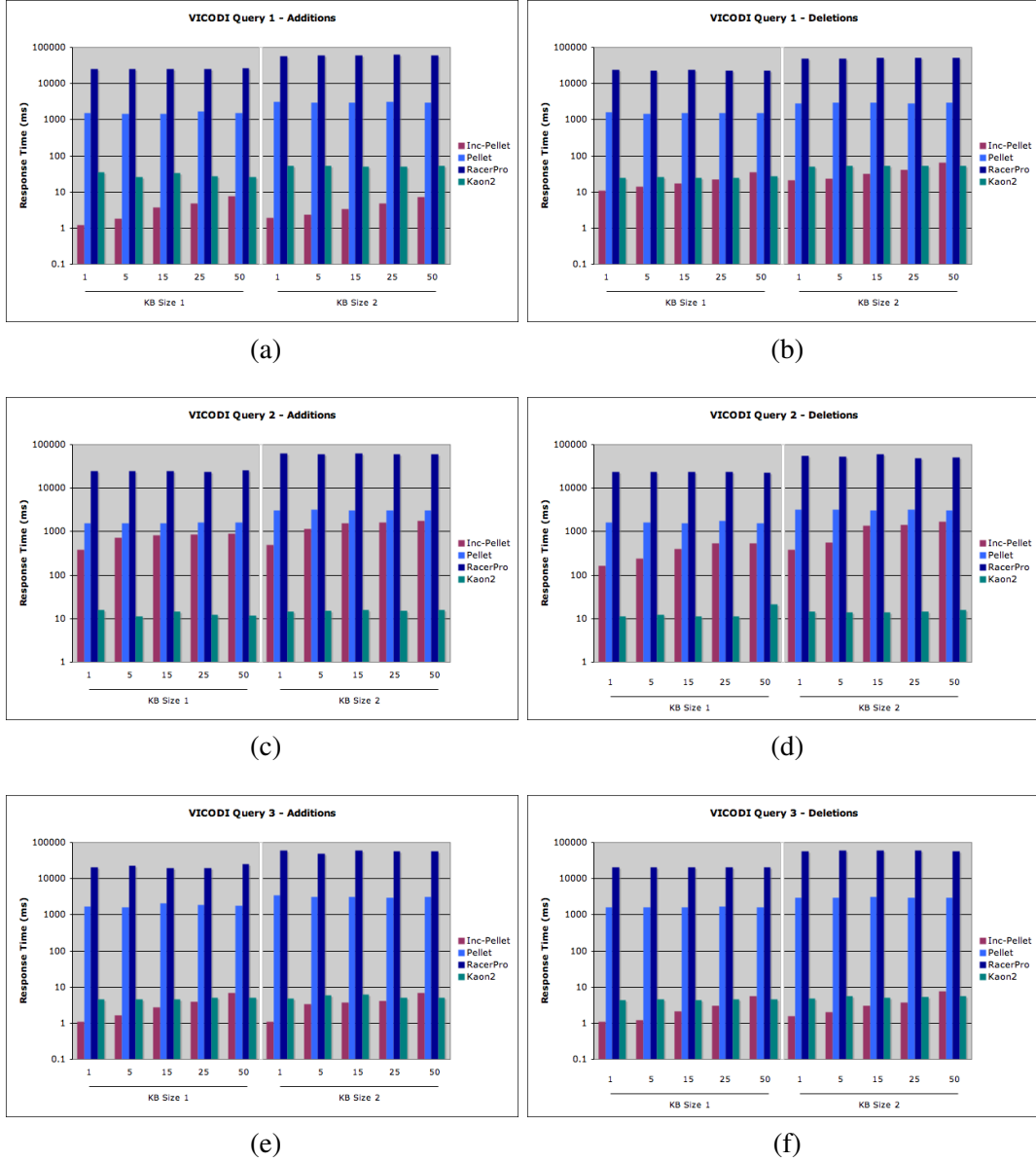


Fig. 2. VICODI Results (a) query 1 - additions (b) query 1 - deletions (c) query 2 - additions (d) query 2 - deletions (e) query 3 - additions (f) query 3 - deletions.

The queries used for the SEMINTEC datasets are provided below. Similar to the VICODI queries, the SEMINTEC queries have been used previously in literature to evaluate DL query answering and were suggested by the ontology authors.

- (1) $(x) \leftarrow Man(x)$
- (2) $(x, y, z) \leftarrow Man(x) \wedge Gold(y, x) \wedge Region(z) \wedge isCreditCardOf(y, x) \wedge livesIn(x, z)$

Figure 3 presents the response times for both queries. Pellet, RacerPro, and KAON2 exhibit similar characteristics as for the VICODI ontologies. Further, the algorithm developed in this paper again demonstrates dramatic performance improvements

| KB / Query | 1 (%) | 5 (%) | 15 (%) | 25 (%) | 50 (%) |
|------------|-------------|-------------|-------------|-------------|-------------|
| VIC-1 / 1 | 1.6 (.009) | 7.02 (.04) | 20.7 (.12) | 34.4 (.2) | 65.6 (.38) |
| VIC-2 / 1 | 1.7 (.01) | 7.4 (.02) | 21 (.06) | 35.4 (.1) | 67.6 (.19) |
| VIC-1 / 2 | 7,128 (42) | 11,440 (67) | 11,553 (68) | 11,560 (68) | 11,536 (68) |
| VIC-2 / 2 | 8,375 (24) | 19,358 (57) | 22,958 (67) | 23,062 (68) | 23,126 (68) |
| VIC-1 / 3 | 1.9 (.01) | 7.16 (.04) | 21.6 (.12) | 34.9 (.2) | 72.8 (.43) |
| VIC-2 / 3 | 1.9 (.005) | 7.2 (.02) | 21.5 (.06) | 35.3 (.1) | 71.6 (.2) |
| SEM-1 / 1 | 1.5 (.008) | 7.2 (.04) | 21.8 (.12) | 36.8 (.2) | 71.8 (.4) |
| SEM-2 / 1 | 1.55 (.004) | 7.5 (.02) | 22.1 (.06) | 37 (.1) | 73.4 (.2) |
| SEM-1 / 2 | 6.2 (.03) | 27.2 (.15) | 95.4 (.53) | 147.6 (.82) | 206.6 (1.1) |
| SEM-2 / 2 | 22 (.06) | 27.8 (.07) | 86.6 (.24) | 157 (.43) | 243 (.67) |
| LUBM-1 / 1 | 0 (0) | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| LUBM-2 / 1 | 9.97 (.02) | 17 (.04) | 17 (.04) | 17 (.04) | 17 (.04) |
| LUBM-1 / 2 | 1.4 (.008) | 7.3 (.04) | 22.3 (.13) | 36.7 (.2) | 74.1 (.45) |
| LUBM-2 / 2 | 2.8 (.007) | 28.5 (.07) | 80 (.2) | 133.8 (.3) | 263.2 (.7) |
| LUBM-1 / 3 | 14.6 (.09) | 50.2 (.3) | 138.8 (.8) | 216 (1.3) | 404 (2.4) |
| LUBM-2 / 3 | 2 (.005) | 7.2 (.01) | 19.5 (.05) | 32.7 (.08) | 67.6 (.18) |
| UOB-1 / 1 | 1.9 (.007) | 29 (.11) | 66.8 (.26) | 111.4 (.4) | 238.3 (.9) |
| UOB-2 / 1 | 1.5 (.002) | 21.7 (.04) | 58.8 (.1) | 98.3 (.1) | 201.6 (.3) |
| UOB-1 / 2 | 3.2 (.01) | 17.6 (.06) | 52.3 (.2) | 85.7 (.33) | 174.3 (.6) |
| UOB-2 / 2 | 3.2 (.006) | 12.3 (.02) | 55.1 (.1) | 87.1 (.16) | 157.1 (.3) |
| UOB-1 / 3 | .9 (.003) | 1.1 (.004) | 35.1 (.13) | 25 (.09) | 80 (.3) |
| UOB-2 / 3 | .04 (.0005) | .3 (.0006) | 10.5 (.02) | 20.8 (.04) | 46.4 (.08) |

Table 4

Addition update candidate sizes for different queries and update sizes for each test-suite ontology. Each column shows the total number of candidates for the specific update size (indicated in the top column) and the percentage of the total KB that this constitutes (shown in parenthesis).

for all update sizes. In many cases, the results are below 10ms. Table 4 presents the actual number of candidates considered on average for addition updates. Again, there is a dramatic reduction in the portion of the KB that must be considered for the query after the updates. The average incremental consistency checking times for this ontology can be found in Chapter 5 of [26]; similar to VICODI, they were on average below 10 milliseconds for addition and deletion updates.

As discussed, the LUBM benchmark includes a set of 14 queries for performance analysis of DL systems. The results of the following 3 queries²² are presented.

- (1) $(x, y, z) \leftarrow GraduateStudent(x) \wedge University(y) \wedge Department(z) \wedge memberOf(x, z) \wedge subOrganization(z, y) \wedge undergraduateDegreeFrom(x, y)$
- (2) $(x) \leftarrow Student(x)$
- (3) $(x, y, z) \leftarrow Student(x) \wedge Faculty(y) \wedge Course(z) \wedge advisor(x, z) \wedge takesCourse(x, z) \wedge teacherOf(y, z)$

²² Note that these correspond to LUBM queries 2, 6, 9 respectively.

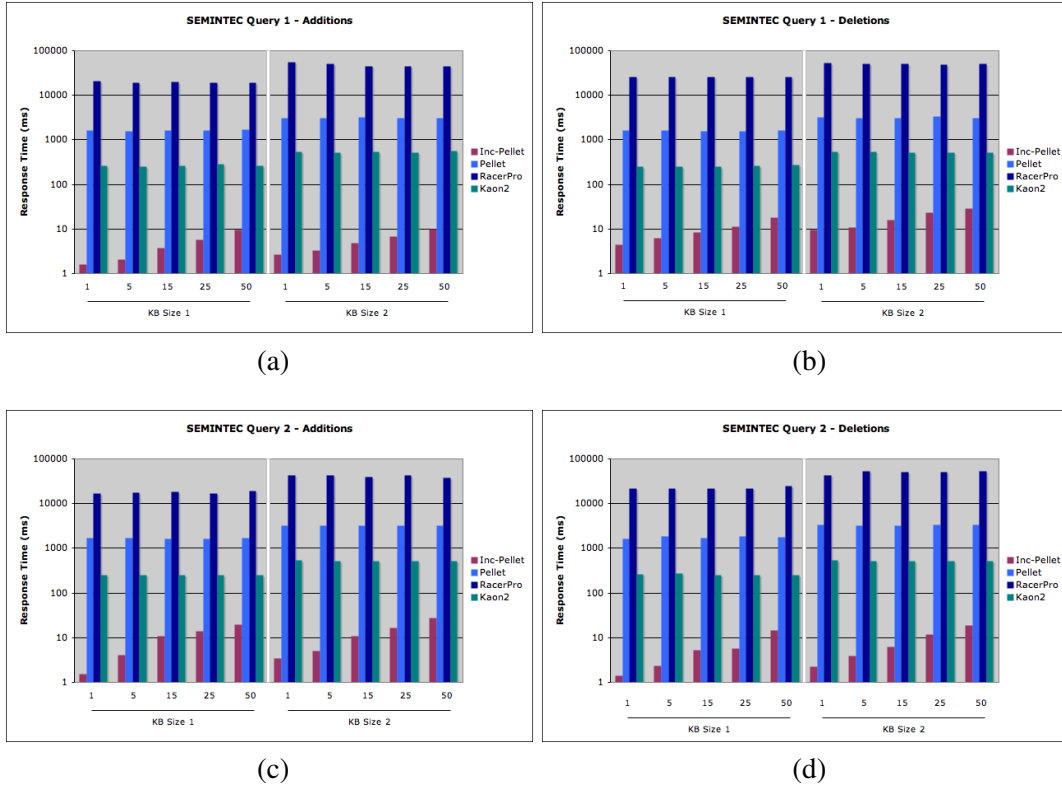


Fig. 3. SEMINTEC Results (a) query 1 - additions (b) query 1 - deletions (c) query 2 - additions (d) query 2 - deletions.

This subset was selected because the results for the remaining queries are similar. Figure 4 presents the results for the queries. The incremental algorithm again demonstrates dramatic performance improvements, as the portion of the KB that must be considered after the update is very small (shown in Table 4). In the queries, the approach typically exhibits response times in the 10s of milliseconds and in all cases shows orders of magnitude improvements over the regular version of Pellet and RacerPro. Further, the approach outperforms KAON2 in many cases. An interesting observation can be made from the number of candidates under additions presented in Table 4. In particular, for query 1 for the smaller of the LUBM datasets, there are 0 candidates. In this case, there are no answers for the query in the entire KB, and when performing the syntactic mapping to take into account the query impact, the initial candidates can never be mapped into the cached completion graph. In the case of the larger dataset, there are only a few mappings in the entire KB, which are located during the approach. Another interesting observation can be made regarding query 3; specifically, the number of candidates is actually smaller on average in the experiments involving the larger of the two datasets. This can be explained by the fact that in the larger dataset there is a larger number of individuals which do not participate in a concept guide path (for the negated, rolled-up query concept) with some other individual, and there is a larger number of individuals that are not mappable into the query (under query impact). Thus, when the random updates are selected, there is a greater chance that these individuals will

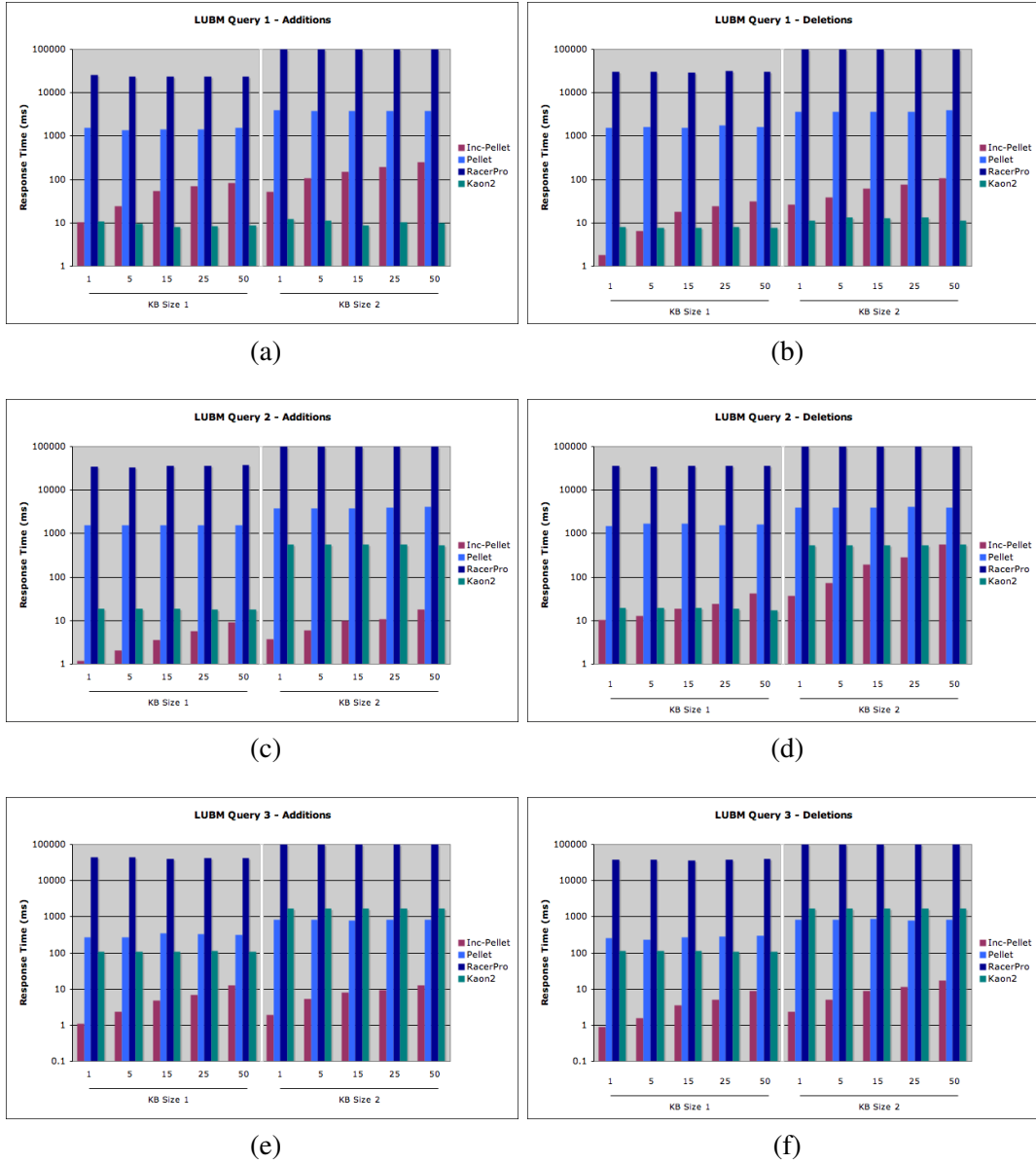


Fig. 4. LUBM (a) query 1 - additions (b) query 1 - deletions (c) query 2 - additions (d) query 2 - deletions (e) query 3 - additions (f) query 3 - deletions.

be selected. The average incremental consistency checking times for addition updates in this ontology were below 10 milliseconds on average and in the 10s of milliseconds for deletions (again, the interested reader is referred to Chapter 5 of [26]).

As with the case for LUBM, the UOB benchmark provides a suite of queries, and the following queries²³ have been used:

$$(1) (x) \leftarrow \text{UndergraduateStudent}(x) \wedge \text{takesCourse}(x, \text{Course}0)$$

²³ Note that these correspond to UOB queries 1, 2, 4 respectively.

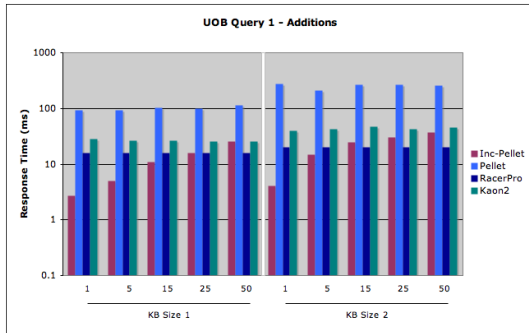
- (2) $(x) \leftarrow \text{Employee}(x)$
 (3) $(x, y) \leftarrow \text{Publication}(x) \wedge \text{Faculty}(y) \wedge \text{isMemberOf}(y, \text{University0}) \wedge \text{publicationAuthor}(x, y)$

These queries were selected as the results for the other queries are similar. Figure 5 presents the results for the queries. The response times of all reasoners were comparable in query 1. The technique presented in this paper resulted in a dramatic reduction in the candidates considered (shown for addition updates in Table 4), and the incremental version of Pellet outperformed re-running the query from scratch. In queries 2 & 3, the developed approach substantially outperforms the other reasoners, demonstrating orders of magnitude performance improvement and generally response times in the 10s of milliseconds. Lastly, the average incremental consistency checking times for this ontology were below 20 of milliseconds (see Chapter 5 of [26]) for addition updates and below 60 milliseconds for deletion updates.

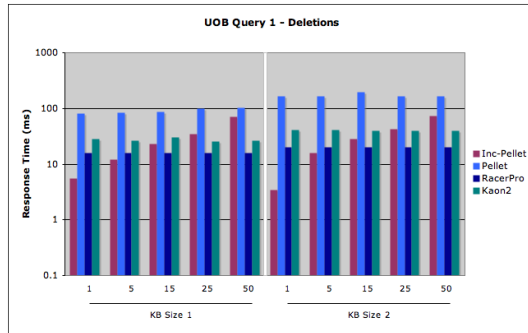
7 Related Work

Early syndication systems primarily relied on subject-based keywords in order to match user interests with published documents/data [37,49]. Following this, approaches allowed attribute-value pairs to be associated with published content (e.g., [1,15]). Recently, there has been interest in utilizing XML-based approaches (e.g., see [3,7,21,11]), in which published documents/data are represented in XML and subscription requests are specified using an XML query/path language (e.g., XPath [7]). There has also been interest in using formal knowledge representation languages for representing published contents (e.g., [9,40,46]); typically such approaches have used RDF as the formalism for encoding publications (e.g., [40,46]). In such an approach, RDF graph-based query languages (typically triple patterns) are used to represent subscription requests and matching publications with subscriptions reduces to triple pattern matching. Additionally, in many approaches RDFS is also utilized to describe domain ontologies which the published RDF content adheres to, allowing simple semantic inferences to be made. There has also been work on addressing the scalability of all of the previously mentioned approaches by leveraging distributed syndication architectures [5,49,11,50,8].

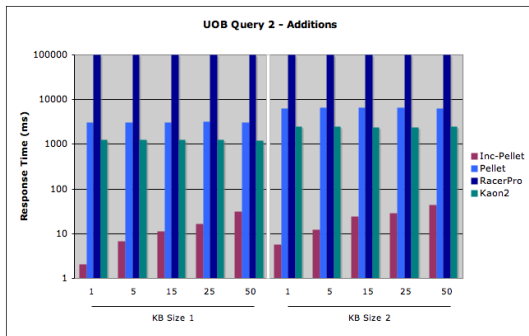
Recently, there has been interest in leveraging DL reasoning in applications which are similar to syndication systems. In [45,33], a DL-based approach for Web service matching is presented, in which DL concepts are used to represent both service subscription requests as well as published documents/data. Matching reduces to determining if published concepts and subscriptions are logically equivalent, subsume one another, or are not compatible. While empirical results demonstrate accepted performance times (~ 20 ms) for matching new subscription requests with a fixed set of published documents, processing a large amount of incoming published con-



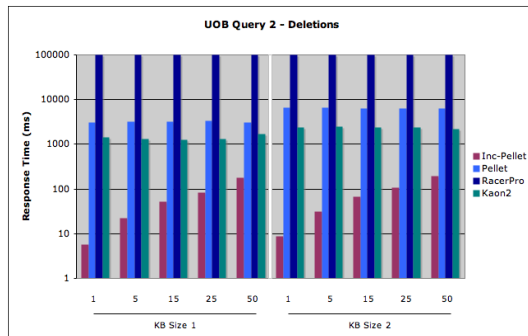
(a)



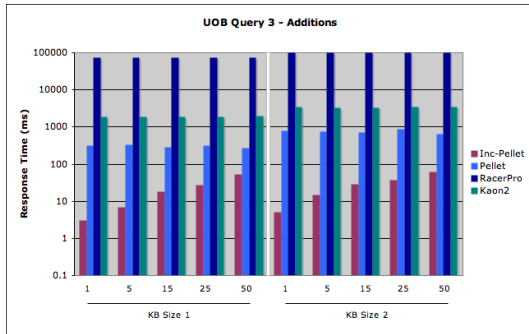
(b)



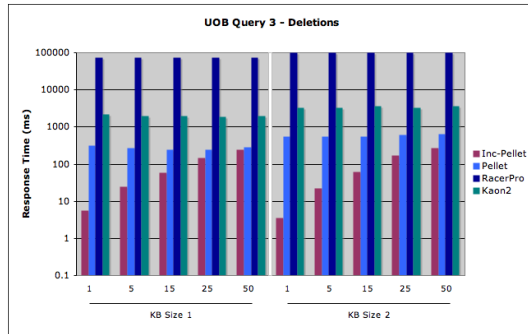
(c)



(d)



(e)



(f)

Fig. 5. OUB (e) query 1 - additions (f) query 1 - deletions (g) query 2 - additions (h) query 2 - deletions (i) query 3 - additions (j) query 3 - deletions.

tent is still problematic (~ 10 s of seconds) [45,33]. [22,23] presents an agent-based document retrieval system (which is essentially a publish/subscribe application) in which published contents are represented as ABox assertions and subscription requests as a DL concept (viewed as an instance retrieval query). Therefore, matching is reduced to instance retrieval of the subscription concept. Given the performance issues of using such an approach (i.e., response times in the 10s of seconds), the authors introduce two optimizations for more effective incremental instance retrieval (discussed later).

In general our work is based a more expressive mechanism for representing published contents. This allows the use of automated reasoning procedures to infer matches not found using syntactic approaches (keyword, attribute-value pair, and/or XML) and simpler semantic approaches (e.g., RDF/S). Further, the related DL-based techniques either take a different approach for representing published contents and subscriptions requests (e.g., [45,33]) or assume a simpler subscription format with only atomic concepts (e.g., [22,23]).

While there has been substantial work on optimizing reasoning services for description logics, the topic of reasoning through evolving DL knowledge bases remains relatively unaddressed. There are a few notable exceptions; [23] presents two approaches for optimizing query answering, namely inducing a partial ordering upon all queries (assumed to be concept retrieval queries) and disregarding previous individuals that satisfied queries. This is directly related to the techniques presented in this paper, however, we present novel techniques to prune the individuals in the KB that must be considered for queries after updates are developed. Further, our approach supports conjunctive queries. There has also been recent work on optimizing classification of DL KBs in the presence of arbitrary TBox changes [20,39]. While related, the work presented here proposes different techniques and addresses different reasoning services, as they are required for the syndication framework. We also point out that there has been substantial work on incremental query and view maintenance in databases (e.g., [6,41,42]) and rule-based systems (e.g., Datalog [12,13]). While related, our work addresses a more expressive formalism.

There has additionally been extensive work in Truth Maintenance Systems (TMSs) for logical theories (e.g., [10,14]). When comparing the contributions of this paper with TMSs, it can be seen that there are some similarities. However, the approaches presented in this paper address optimizing the detection of non-entailments in the presence of updates, which is not usually addressed in TMSs. Further, TMSs traditionally only support propositional logic, however, in this paper a more expressive formalism is supported.

8 Conclusion

In this paper, we have formalized an OWL-based syndication framework in which DL reasoning is the means for matching newly published information with subscription requests. We then addressed one of the main limitations with such a syndication framework, namely efficiently matching new information with registered subscriptions. To this end, we presented a novel algorithm for incremental query answering. Our preliminary results demonstrate orders of magnitude performance improvements over state-of-the-art DL reasoners. Further, the technique provides a matching mechanism that can scale to hundreds and in some cases thousands of subscriptions under high-demand publish frequencies similar to that encountered in the financial domain (e.g., the Dow Jones Newswires has up to 12,000 publications per day ~ 8.3 per minute²⁴). Future work includes investigating extensions to the framework such as supporting TBox updates in publications, as well as developing additional optimizations for the reasoning techniques required for the framework. The later includes extending the techniques to larger portions of OWL (and beyond) and lifting the restrictions currently imposed in the approach. Additionally, we plan to investigate more advanced mechanisms to recover from inconsistencies that can occur from new publications; currently, we are working on developing trust-based revision techniques for OWL DL KBs [19].

We would like to thank Jennifer Golbeck, Yarden Katz, Bijan Parsia, Evren Sirin, and Taowei Wang for all of their contributions to this work. This work was supported by grants from Fujitsu, Lockheed Martin, NTT Corp., Kevric Corp., SAIC, the National Science Foundation, the National Geospatial - Intelligence Agency, DARPA, US Army Research Laboratory, and NIST.

References

- [1] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, T. D. Chandra, Matching events in a content-based subscription system, in: Symposium on Principles of Distributed Computing, 1999.
- [2] C. E. Alchourrón, P. Gärdenfors, D. Makinson, On the logic of theory change: Partial meet contraction and revision functions., *Journal of Symbolic Logic* 50 (2) (1985) 510–530.
- [3] M. Altinel, M. J. Franklin, Efficient filtering of XML documents for selective dissemination of information, in: *The VLDB Journal*, 2000.
- [4] F. Baader, B. Hollunder, Embedding defaults into terminological representation systems, *J. Automated Reasoning* 14 (1995) 149–180.

²⁴ Source: http://www.dj.com/Products_Services/ElectronicPublishing/DJNewswires.htm

- [5] G. Banavar, T. D. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, D. C. Stur, An efficient multicast protocol for content-based publish-subscribe systems, in: ICDCS '99: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, IEEE Computer Society, Washington, DC, USA, 1999.
- [6] J. A. Blakeley, P.-A. Larson, F. W. Tompa, Efficiently updating materialized views, in: Proc. of SIGMOD '86: ACM SIGMOD International Conference on Management of Data, 1986.
- [7] C. Y. Chan, P. Felber, M. N. Garofalakis, R. Rastogi, Efficient filtering of XML documents with XPath expressions, The VLDB Journal 11 (2002) 354–379.
- [8] P. A. Chirita, S. Idreos, M. Koubarakis, W. Nejdl, Publish/subscribe for rdf-based p2p networks, in: Proceedings of the 1st European Semantic Web Symposium. (2004), 2004.
- [9] M. Cilia, C. Bornhvd, A. P. Buchmann, Cream: An infrastructure for distributed, heterogeneous event-based applications, in: Proceedings of the International Conference on Cooperative Information Systems, 2003.
- [10] J. de Kleer, An assumption-based TMS, Artif. Intell. 28 (2) (1986) 127–162.
- [11] Y. Diao, S. Rizvi, M. Franklin, Towards an internet-scale xml dissemination service, in: Proceedings of VLDB2004, August 2004., 2004.
- [12] G. Dong, J. Su, R. W. Topor, Nonrecursive incremental evaluation of datalog queries, Annals of Mathematics and Artificial Intelligence 14 (2-4) (1995) 187–223.
- [13] G. Dong, R. W. Topor, Incremental evaluation of datalog queries, in: Proc. of the 4th Int. Conference on Database Theory, 1992.
- [14] J. Doyle, A truth maintenance system, Readings in nonmonotonic reasoning (1987) 259–279.
- [15] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, D. Shasha, Filtering algorithms and implementation for very fast publish/subscribe systems, SIGMOD Record (ACM Special Interest Group on Management of Data) 30 (2) (2001) 115–126.
- [16] G. Flouris, D. Plexousakis, G. Antoniou, On applying the agm theory to dls and owl, in: 4th International Semantic Web Conference (ISWC 2005), 2005.
- [17] B. Glimm, I. Horrocks, C. Lutz, U. Sattler, Conjunctive query answering for the description logic *SHIQ*, in: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007), 2007.
- [18] B. Glimm, I. Horrocks, U. Sattler, Conjunctive query entailment for *SHOQ*, in: Proc. of the 2007 Description Logic Workshop (DL 2007), vol. 250, 2007.
- [19] J. Golbeck, C. Halaschek-Wiener, Trust-based revision for expressive web syndication, Journal of Logic and Computation. To Appear.

- [20] B. C. Grau, C. Halaschek-Wiener, Y. Kazakov, Historymatters: Incremental ontology reasoning using modules, in: Proceedings of the 6th International Semantic Web Conference, 2007.
- [21] A. K. Gupta, D. Suci, Stream processing of xpath queries with predicates, in: SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM Press, New York, NY, USA, 2003.
- [22] V. Haarslev, R. Moller, Description logic systems with concrete domains: Applications for the semantic web, in: Int. Workshop on KR meets Databases, 2003., 2003.
- [23] V. Haarslev, R. Möller, Incremental query answering for implementing document retrieval services, in: Proceedings of the International Workshop on Description Logics (DL-2003), Rome, Italy, September 5-7, 2003.
- [24] C. Halaschek-Weiner, J. Hendler, Toward expressive syndication on the web, in: Proceedings of the 16th World Wide Web Conference, 2007.
- [25] C. Halaschek-Wiener, B. Parsia, E. Sirin, A. Kalyanpur, Description logic reasoning with syntactic updates, in: In Proc. of the 5th Int. Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2006), 2006.
- [26] F. C. Halaschek-Wiener, Expressive syndication on the web using a description logic based approach, Ph.D. thesis, University of Maryland, College Park, <http://www.mindswap.org/~chris/publications/FCHW-Thesis.pdf> (2007).
- [27] I. Horrocks, U. Sattler, A tableaux decision procedure for SHOIQ, in: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Morgan Kaufman, 2005.
- [28] I. Horrocks, U. Sattler, S. Tobies, Practical reasoning for expressive description logics, in: Proc. of the 6th Int. Conference on Logic for Programming and Automated Reasoning (LPAR'99), No. 1705 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 1999.
- [29] I. Horrocks, S. Tessaris, A conjunctive query language for description logic aboxes, in: National conference on artificial intelligence (AAAI 2000), 2000.
- [30] Z. Ives, A. Levy, D. Weld, Efficient evaluation of regular path expressions on streaming xml data, in: Univ. of Washington Tech. Rep. CSE000502., 2000.
- [31] A. Kalyanpur, Debugging and repair of owl ontologies, in: Ph.D. Dissertation, University of Maryland, College Park, 2006, <http://www.mindswap.org/papers/2006/AdityaThesis-DebuggingOWL.pdf>.
- [32] L. V. S. Lakshmanan, S. Parthasarathy, On efficient matching of streaming XML documents and queries, in: Extending Database Technology, 2002.
- [33] L. Li, I. Horrocks, A software framework for matchmaking based on semantic web technology, in: Proceedings of the Twelfth International World Wide Web Conference (WWW 2003), 2003., 2003.
- [34] H. Liu, C. Lutz, M. Milicic, F. Wolter, Updating description logic aboxes, in: International Conference of Principles of Knowledge Representation and Reasoning(KR), 2006.

- [35] A. O. Mendelzon, P. T. Wood, Finding regular simple paths in graph databases, in: VLDB '89: Proceedings of the 15th international conference on Very large data bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.
- [36] B. Motik, U. Sattler, A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes, in: M. Hermann, A. Voronkov (eds.), Proc. of the 13th Int. Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006), vol. 4246 of LNCS, Springer, Phnom Penh, Cambodia, 2006.
- [37] B. Oki, M. Pfluegl, D. Skeen, The information bus: An architecture for extensible distributed systems, in: In Proc. 14th SOSP, 1993.
- [38] M. M. Ortiz de la Fuente, D. Calvanese, T. Eiter, Data complexity of answering unions of conjunctive queries in shiq, in: Proc. of the 2006 Description Logic Workshop (DL 2006), CEUR Electronic Workshop Proceedings, <http://ceur-ws.org/>, 2006.
- [39] B. Parsia, C. Halaschek-Wiener, E. Sirin, Towards incremental reasoning through updates in owl-dl, in: Reasoning on the Web - Workshop at 15th International World Wide Web Conference, 2006.
- [40] M. Petrovic, H. Liu, H.-A. Jacobsen, Cms-topss: Efficient dissemination of rss documents, in: VLDB '05: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, 2005.
- [41] M. Stonebraker, Implementation of integrity constraints and views by query modification, in: SIGMOD '75: Proc. of the 1975 ACM SIGMOD international conference on Management of data, New York, NY, USA, 1975.
- [42] D. B. Terry, D. Goldberg, D. Nichols, B. M. Oki, Continuous queries over append-only databases, in: Proc. of the Intl. Conf. on Management of Data, 1992.
- [43] S. Tessaris, Questions and answers: reasoning and querying in description logic, Ph.D. thesis, University of Manchester (2001).
- [44] D. Tsarkov, I. Horrocks, Efficient reasoning with range and domain constraints, in: Proc. of the Int. Description Logic Workshop (DL 2004), 2004.
- [45] M. Uschold, P. Clark, F. Dickey, C. Fung, S. Smith, S. U. M. Wilke, S. Bechhofer, I. Horrocks, A semantic infosphere, in: D. Fensel, K. Sycara, J. Mylopoulos (eds.), Proc. of the 2003 International Semantic Web Conference (ISWC 2003), No. 2870 in Lecture Notes in Computer Science, Springer, 2003.
- [46] J. Wang, B. Jin, J. Li, An ontology-based publish/subscribe system, in: Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, New York, NY, USA, 2004.
- [47] T. D. Wang, B. Parsia, J. Hendler, A survey of the web ontology landscape, in: Proceedings of the International Semantic Web Conference (ISWC2006, 2006.
- [48] M. Winslett, Updating logical databases, in: Updating Logical Databases, Cambridge University Press, 1990.

- [49] T. W. Yan, H. Garcia-Molina, The SIFT information dissemination system, *ACM Transactions on Database Systems* 24 (4) (1999) 529–565.
- [50] E. Yoneki, J. Bacon, Distributed multicast grouping for publish/subscribe over mobile ad hoc networks, in: *Wireless Communications and Networking Conference*, 2005.