

Expressive Logic-Based Syndication on the Web

(UMIACS Technical Report)

Christian Halaschek-Wiener and James Hendler
The University of Maryland
Department of Computer Science
College Park, Maryland
{halasche,hendler}@cs.umd.edu

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 expressive Web ontologies and logic-based 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 continuous query answering for a large subset of the Web Ontology Language (OWL); specifically, we formally define continuous queries for OWL knowledge bases and present a novel algorithm for continuous query answering in a large subset of this language. Lastly, an evaluation of the query approach is shown, demonstrating its effectiveness for syndication purposes.

1 Introduction

Web-based syndication systems have attracted a great amount of attention in recent years. In typical syndication frameworks, users register their subscription requests with syndication brokers; similarly, content publishers register their feeds with syndication brokers (see Figure 1). It is then the broker's task to match newly published information with registered subscriptions. 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 for describing their interests and published content, enabling more accurate dissemination. Through the years there has been a transition from keyword-based approaches (e.g., [26]) to attribute-value pairs (e.g., [1]) and more recently to XML (e.g., [2, 6]). Given the limited knowledge modeling expressivity of XML (and XML Schema), there has been interest in using RDF [22] for syndication purposes (e.g., [5, 33]). RDF has even been adopted as the standard representation format of RSS 1.0.

Today's syndication approaches still provide relatively weak expressive power from a modeling perspective (i.e., XML and RDF are inexpressive modeling languages) and provide very little automated reasoning support. However, if a more expressive 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 grained control for filtering [32]. Additionally, reasoning can be utilized for discovering subscription matches not found using traditional syntactic syndication approaches.

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 [12, 32, 24]. 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 increased expressivity of an OWL-based syndication approach, consider the following example related to the financial domain. Assume that a stock trader is interested in information contained in news articles (or collections of articles) that discuss news about companies that will make their stocks volatile (i.e., they become risky investments). In particular, assume that the trader

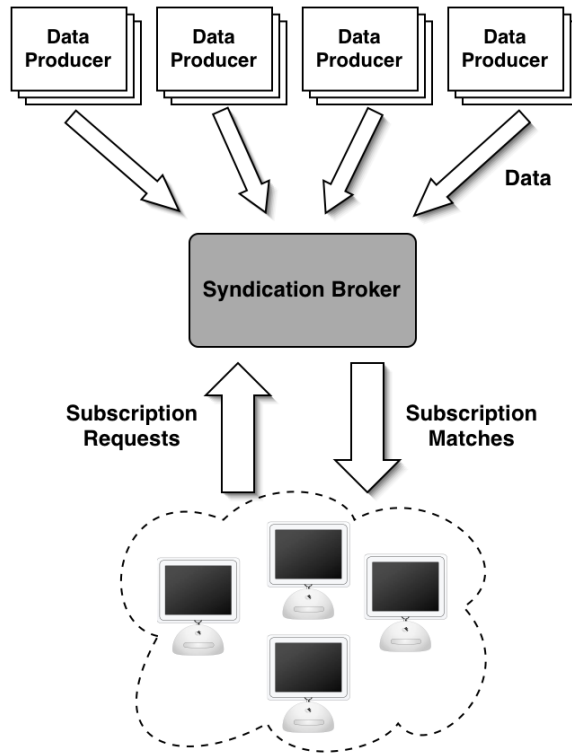


Figure 1: Basic Syndication Architecture

is interested in any *RiskyCompany* that he or she defines to include companies that had their credit downgraded by either Moodys or S&P credit agency and exists on some sell ratings list of a financial institution. 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. However, more complex logical definitions and automatic classification of objects cannot be supported; therefore, an XML-based approach cannot accommodate the previous example. If we consider an RDF-based approach, then a syndication broker can model the financial domain using RDF Schema; therefore, slightly more complex subscription matches can be obtained, as one can logically infer that a company is a *RiskyCompany* (e.g., based on subclass relationships). However, in an RDF-based approach, complex logical definitions, such as the previously mentioned *RiskyCompany*, are not definable. In contrast, in an OWL-based approach, such expressivity is easily provided. For example, *RiskyCompany* is defined in

Table 1 (turtle syntax).

```
:RiskyCompany a owl:Class;
  owl:intersectionOf (
    [ a owl:Restriction; owl:onProperty :onRecommendation;
      owl:someValuesFrom :SellList ]
    [ a owl:Restriction; owl:onProperty :downgradedBy;
      owl:someValuesFrom [ owl:oneOf ( :SandP :Moodys ) ] ]
  ) .
:movedToJunk a owl:ObjectProperty;
  rdfs:subPropertyOf :downgradedBy .
:onRecommendation a owl:ObjectProperty;
  owl:InverseOf :hasRecommendation .
```

Table 1: Illustration of expressivity in OWL-based syndication.

The definitions that the property *movedToJunk* is a sub-property of *downgradedBy*, and *onRecommendation* is the inverse of *hasRecommendation*, are included in Table 1 as they are used later in the paper.

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 [32, 24, 12]. This is an issue in domains such as the syndication of financial news feeds 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. The consistency of the KB must be ensured; queries must be re-evaluated; etc. This negatively impacts the performance results of existing DL-based syndication approaches, as performance times are in the tens of seconds. An additional limitation of OWL-based syndication approaches is related to the infancy of the underlying architectures investigated to date; in particular, these architectures have not been investigated in great depth or fully formalized.

In this paper we address both of the previously discussed shortcomings with OWL-based syndication systems. In particular, we first formalize a DL-based syndication framework. We then address the scalability of DL reasoning for the purpose of syndication. First, we present a technique for incremental consistency checking for a substantial portion of OWL. Then, we address the issue of continuous query answering over OWL KBs that are being updated, primarily focusing on

reducing the size of the KB that must be considered as candidate query bindings. This effectively allows a smaller subset of the KB to be considered for possible subscription matches. The techniques we present are applicable to queries with at least one distinguished variable (i.e., must be bound to a named individual) allowing the query to be rolled-up into a distinguished variable and only contain simple roles (i.e., no transitive roles or super-roles of a transitive role). Further, the approach supports the description logic *SHI* (a large subset of OWL) with the restriction the KB unfoldable (i.e., acyclic). Lastly, an evaluation of the incremental reasoning techniques is provided, demonstrating their effectiveness for OWL-based syndication. Our initial results show dramatic improvements, demonstrating that the matching approach is more practical for syndication purposes.

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 and Description Logics

The W3C-approved Web Ontology Language (OWL) is the recommended standard for the 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* [15], 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 of *SHI* (semantics can be found in [15]).

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)¹. We write \top and \perp to abbreviate

¹See [15] for a precise definition of simple roles.

$C \sqcup \neg C$ 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. To avoid considering the role R^- we introduce the function $\text{Inv}(R)$, which returns the inverse of a role R . Additionally, for a role hierarchy \mathcal{R} let the symbol $\sqsubseteq_{\mathcal{R}}$ denote the reflexive transitive closure of \sqsubseteq on $\mathcal{R} \cup \{\text{Inv}(R_1) \sqsubseteq \text{Inv}(R_2) \mid R_1 \sqsubseteq R_2 \in \mathbf{R}\}$. We also use $R_1 \equiv_{\mathcal{R}} R_2$ as an abbreviation for $R_1 \sqsubseteq_{\mathcal{R}} R_2$ and $R_2 \sqsubseteq_{\mathcal{R}} R_1$. We define the function $\text{Tr}(R, \mathcal{R})$ that returns *true* if R is a transitive role; otherwise the function returns *false*. A role R_1 is considered *simple* with respect to \mathbf{R} if $\text{Tr}(R_2, \mathcal{R}) = \text{false}$ for all $R_2 \sqsubseteq_{\mathcal{R}} R_1$.

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 [15].

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 an 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} . We also introduce the following notation: denote by $\text{Mod}(\mathbf{K})$ the set of all models for \mathbf{K} . Additionally, given a *SHI* concept C , denote by $\text{Depth}(C)$ the maximum modal depth for C (i.e., the maximum nesting depth of quantifiers).

2.2 Conjunctive ABox Queries

In this section, we provide a brief overview of conjunctive ABox queries (query, for short) for description logics. A query Q consists of a conjunction of ABox assertions of the form $C(a)$ or $R(a, b)$ (see [19] for a precise definition), in which variables can be used in place of individuals and are considered as existentially quantified (the set of variable names, denoted $V(Q)$, is assumed to be distinct from the individual names, \mathbf{I}). Query answering is the task for determining if Q is a logical consequence of the KB \mathbf{K} (denoted $\mathbf{K} \models Q$); that is, determining if for all models \mathcal{I} of \mathbf{K} , $\mathcal{I} \models Q$. As query retrieval is addressed in this work, we briefly introduce the following notation (adopted from [19]): $\langle x_1, \dots, x_n \rangle \leftarrow Q$ 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. These variables are referred to as *distinguished* variables, denoted $DV(Q)$. The *answer set* of a query $\langle x_1, \dots, x_n \rangle \leftarrow Q$ w.r.t. to \mathbf{K} is the set n -ary tuples defined by the following:

$$\{\langle a_1, \dots, a_n \rangle \in \mathbf{I}_{\mathbf{K}}^n \mid \mathbf{K} \models Q[x_1/a_1, \dots, x_n/a_n]\}$$

where $Q[x/a]$ represents the query, Q , with all occurrences of variable x substituted by the individual name a . Lastly, we note that 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 [10]. We additionally introduce the following notation: given query Q , let $Con(Q)$, $Rol(Q)$ denote the set of concepts and roles in Q respectively. In the remainder, we assume all queries contain at least one distinguished variable and can be rolled-up [18, 19] into some distinguished variable. Given a query Q , with abuse of notation, we denote by $Depth(Q)$ the $Depth(C)$, where C is the concept obtained by rolling-up the query into a distinguished variable and then fully unfolding the negation of the concept obtained.

2.3 Tableau Algorithms

DL tableau-based algorithms decide the consistency of an ABox \mathbf{A} with respect to a TBox \mathbf{T} and RBox \mathbf{R} by trying to construct (an abstraction of) a common model for \mathbf{A} , \mathbf{T} , and \mathbf{R} , called a *completion graph* [15]. 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. A node, x , contains a clash if a contradiction exists in its label (e.g., $C, \neg C \in \mathcal{L}(x)$) or between two nodes (equality and/or inequality). It is noted that the tableau algorithm can be saturated such that all possible completions of a KB are found (corresponding to all models). We lastly introduce the following notation: denote by $Comp(K)$ the set of all complete, clash-free completions of K (i.e., all models); additionally, given completion graph G , denote by $Roots(G)$ the subset of G containing root nodes and their labels, as well edges and edge labels between root nodes.

3 Syndication Framework

In this section we formally define the generic DL-based syndication framework proposed in this work. As in typical syndication systems, we assume syndication brokers deliver relevant information to the appropriate subscription requests. Within this framework, a subscription is comprised of a conjunctive ABox (instance) query, which represents the subscribers interests, and an expiration time (i.e., the number of time units that the subscription is valid). The subscription query can be thought of as a continuous conjunctive query that should be evaluated until the expiration time. Therefore, the query is issued once over a changing ABox whose results set is continuously updated as the ABox changes. Intuitively, the answer set of a continuous conjunctive ABox query at time t is the set of all variable bindings entailed by the KB at time t and can be seen as an extension of the definition of a conjunctive query presented earlier.

Definition 1 (*Continuous Conjunctive ABox Query*) Define a continuous conjunction ABox query Q_c with respect to a DL KB K_t (K at time t) such that it produces results at time t , denoted $Q_c(t)$, as follows:

$$Q_c(t) = \{\langle a_1, \dots, a_n \rangle \in \mathbf{I}_{K_t}^n \mid K_t \models Q_c[x_1/a_1, \dots, x_n/a_n]\}$$

Given this, a *subscription* is then defined as follows:

Definition 2 (*Subscription*) A subscription S is defined as a pair (Q_c, t) , where Q_c is a continuous conjunctive ABox query that is evaluated for t time units.

We denote the continuous query of a subscription as $S(Q_c)$ and the expiration time as $S(t)$. We now define a *subscriber* to be composed of a set of subscriptions and a unique identifier:

Definition 3 (Subscriber) A subscriber Sub is defined to be a pair (s, i) , where s is a set of subscriptions and i is a unique identifier.

Denote a subscriber's set of subscriptions as $\text{Sub}(s)$, and its identifier as $\text{Sub}(i)$. Next, we define a *publisher* to be identified by a unique identifier:

Definition 4 (Publisher) A publisher Pub is defined as being composed of and identified by a unique identifier i .

Additionally, a *publication* is defined to be composed of a set of ABox assertions, the number of time units that the publication is valid, and the identifier of the publisher that produced the information; after the specified time units have passed, it is assumed that the publication is discarded.

Definition 5 (Publication) A publication P is defined as a tuple (α, t, p) , where α is a set of DL ABox assertions that expire after t time units, and p is the identifier of the publisher that produced the publication.

Given a publication P , denote the set of ABox assertions as $P(\alpha)$, the expiration time as $P(t)$, etc. Intuitively, a *syndication broker* maintains a local KB, in which newly published information is integrated. Additionally, the syndication broker maintains the currently registered subscribers (that have associated subscriptions) and publishers. This is formally defined as follows:

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

We denote a syndication broker's subscriptions, publishers, and KB as $B(S)$, $B(P)$, and $B(K_l)$ respectively. After a new publication is received, it is the broker's task to determine the subscribers for which this new information is relevant. Before defining subscription matches, we define a generic *update function* that takes a set of publications and integrates them into the broker's KB. Such a function is necessary for integrating newly published information into a DL KB.

Definition 7 (*Update Function*) Define the update function $update(K, P)$ to take as input a DL KB K and a set of publications P and return a new consistent DL KB K' that is the result of updating K with $P(\alpha)$, for all $P \in P$.

Observe that this update function is generic, as there are many different ways to interpret the update. Such problems have been studied extensively in literature for updating logical KBs (e.g., [34]). We do not impose a particular type of update semantics in the formalization of the syndication framework; rather, we only enforce that the update function result in a consistent KB. This is necessary because if the updated KB is inconsistent, then everything is trivially entailed. In Section 4.1 we define a specific update function, referred to as *syntactic updates*.

Lastly, we define a match for a subscription request. As information (documents) is published from multiple publishers and remains valid in the broker's local KB for varying time, a match for a subscription can actually be a composition of the information from multiple publications; that is, the information provided in multiple publications collectively forms a match for the query. To the authors' knowledge, recent approaches have not investigated such functionality; rather, only information from individually published documents form a match for a given subscription. However, such a capability is beneficial, as information can be considered collectively and form matches not found otherwise.

We additionally distinguish between two types of subscription matches, namely *information matches* and *publication matches*. An information match refers to the individuals bound to the (distinguished) variables of a continuous query representing a subscription; that is, the result returned to the subscriber is actually the query answer rather than the publication(s) responsible for the answer. 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 [21]. 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, return all minimal sets of publications that cause this match to occur; this aligns with the task of selective content-based filtering of publications. It is clear that given an information match, there is a corresponding set of publication matches.

The distinction between these two match types is made as additional computation is needed to derive all the minimal sets of publications responsible for an information match. Further, the type of match required is application dependent; for example, in OWL-based syndication of news feeds, it is clear that publication matches are needed. In contrast, in the financial domain, analysts are generally

interested with the actual information rather than the documents themselves. If we consider our previous example involving the concept *RiskyCompany*, we can observe that analysts are likely to be more interested in the actual instances of *riskycompany*, rather than the articles that discuss them; this is intuitive, as the actual query answer is the actionable information for their purposes (e.g., stock trading). In this work, we address both of these matches; however, our current evaluation focuses on information matches, leaving the remainder as future work. We now define an *information match*:

Definition 8 (*Information Match*) Define a tuple of individuals $\langle a_1, \dots, a_n \rangle$ to be an information match at broker \mathbf{B} for subscription \mathbf{S} at time t , if and only if the following holds:

$$\langle a_1, \dots, a_n \rangle \in \mathbf{B}(\mathbf{I}_{K_t}^n) \wedge \mathbf{B}(K_t) \models \mathbf{S}(Q_c[x_1/a_1, \dots, x_n/a_n])$$

Before defining a publication match, we present the notion of *minimal justifications* for an entailment in DLs, which has been formally investigated in literature [20].

Definition 9 (*Minimal Justification*) [20] Let $\mathbf{K} \models \alpha$, where α is a DL axiom and \mathbf{K} a DL KB. A fragment $\mathbf{K}' \subseteq \mathbf{K}$ is a minimal justification for α in \mathbf{K} if $\mathbf{K}' \models \alpha$ and $\mathbf{K}'' \not\models \alpha$ for every $\mathbf{K}'' \subset \mathbf{K}'$. Denote the set of minimal justifications for $\mathbf{K} \models \alpha$ as $\text{Just}(\mathbf{K}, \alpha)$.

Now we present the definition of a *publication match* which utilizes minimal justifications to define to the publications responsible for an information match:

Definition 10 (*Publication Match*) Let $\langle a_1, \dots, a_n \rangle$ be an information match I at broker \mathbf{B} for subscription \mathbf{S} at time t . Let J be the set of minimal justifications for I :

$$J = \text{Just}(\mathbf{B}(K_t), \mathbf{S}(Q_c[x_1/a_1, \dots, x_n/a_n]))$$

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

$$\forall P \in P (\exists a \in j \wedge a \in P(\alpha)) \wedge \forall a \in j (\exists P \in P \wedge a \in P(\alpha))$$

We conclude this section with a brief example demonstrating a composite match (both information and publication matches), and the framework in general. Assume a syndication broker B is composed of one subscription and two publishers. Additionally, assume that the broker's local KB contains the axioms defined previously in Table 1. The the broker is composed of the following:

$$S = \{S_1\}, \quad P = \{P_1, P_2\}$$

$$K_l = \{ :movedToJunk, :onRecommendation, :RiskyCompany \}$$

Assume subscriber S_1 has registered the following subscription for all instances of the class *RiskyCompany*:

$$(RiskyCompany(x), \infty) \in S_1(s)$$

where ∞ indicates that the subscription does not expire. Additionally assume that P_1 publishes that *BOASellList* (assumed to be an instance of *SellList*) has a sell recommendation for Ford and P_2 publishes Moodys moved Ford credit to junk status. This is formalized as follows:

$$P_{P_1} = (\{ :BOASellList :hasRecommendation :Ford \}, \infty, 1)$$

$$P_{P_2} = (\{ :Moodys :movedToJunk :Ford \}, \infty, 2)$$

where ∞ indicates that the publications do not expire. For ease of exposition, assume that P_{P_1} and P_{P_2} arrive at the broker at time 1 and 2 respectively, and that the update function expands the explicit ABox assertions in the broker's KB with those contained in the publication (see Section 4.1 for further details). When P_{P_1} arrives at the broker, $P_{P_1}(\alpha)$ is integrated into $B(K_l)$, resulting in a updated broker KB K' . It is obvious that at this time the individual Ford will not satisfy the subscription; therefore, there will not be a match for S_1 at time 1. However, when P_{P_2} is published at time 2 and integrated into K' , there is a composite publication match $\{P_{P_1}, P_{P_2}\}$ and an information match Ford for the subscription (due to various OWL inferences).

4 Reasoning for Syndication

As discussed earlier, the main limitation in the proposed syndication framework is related to DL reasoning through incremental changes to the underlying KB. In particular, there exist two main performance bottlenecks; the first is related to ensuring that the broker's KB is consistent after new information is integrated

(i.e., published). When the underlying KB is large, this consistency check can take tens of seconds. The second bottleneck is efficiently re-evaluating registered subscriptions (i.e., continuous queries) after new information is published. While there has been work on providing efficient DL query answering, the notion of incremental or continuous querying answering has been relatively un-addressed. The remainder of this paper addresses these two performance issues, providing a more practical realization of the syndication framework presented in Section 3. However, before addressing these issues, the update function adopted for this work is presented.

4.1 Syntactic Updates

Determining how to interpret updates in logical knowledge bases raises many issues. Such problems have been studied extensively in the literature for updating (propositional and first-order) KBs (e.g., [34]). For the task of syndication, we propose an update function that we refer to as *syntactic updates*, which supports syntactic changes of KB assertions. Intuitively, syntactic updates can be described as an update in which all new assertions are directly added (or removed) to the asserted (base) axioms. For purpose of this work, ABox assertions can take the form of individual equality (e.g., $\{:\text{Ford owl:sameAs :FordMotorCorp}\}$) and inequality assertions (e.g., $\{:\text{Moodyowl:differentFrom :SandP}\}$), concept assertions (e.g., $\{:\text{Ford a :Company}\}$; note that complex concept assertions are possible as well), and role assertions (e.g., $\{:\text{Ford :downgradedBy:Moody}\}$). Formally, this is described as follows:

Definition 11 (*Syntactic Updates*) *Let A be the ABox of an initial KB K . Then, under syntactic updates, updating K with an ABox addition (respectively deletion) α , written as $K + \alpha$ (resp. $K - \alpha$), results in an updated ABox A' such that $A' = A \cup \{\alpha\}$ (resp. $A' = A \setminus \{\alpha\}$). Denote by $K \oplus \alpha$ the syntactic update of K with α .*

This type of update is different when compared to related work in update semantics [25] and belief revision [8] for DLs; however, it is clearly applicable to syndication applications. Further, there have been negative results with respect to other candidate update semantics for DL KBs. In particular, [25] shows that the standard (minimal change) model-based update semantics cannot be represented in the DLs considered in this paper. [8] shows that many DLs, including those considered here, cannot satisfy the rationality postulates proposed in the AGM theory of belief revision. Lastly, as our empirical evaluation demonstrates, practical DL reasoning techniques can be provided under these update semantics as

well. It is clear that under syntactic updates, the resulting KB can be inconsistent after an update; however, as required by Definition 7, the update function must result in a consistent KB. For this work, we assume that if the resulting KB is inconsistent, then the newly published information is rejected (i.e., removed from the KB). While discarding the recent publication may not be the ideal course of action in all syndication systems, addressing this issue further is out of the scope of this paper. However, we plan to address this issue in future work and provide some initial insights in Section 6.

4.2 Incremental Consistency Checking

After newly published information is integrated in the broker’s KB, consistency must be rechecked. As stated earlier, with large ABoxes, checking consistency introduces substantial overhead. In this 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.

To address this issue, we have recently investigated incremental consistency checking in OWL KBs. In particular, in [14] we present an approach for incrementally updating tableau completion graphs under syntactic ABox updates in the description logics *SHIQ* and *SHOQ* [14], which encompass the portion of OWL-DL addressed later in this work. In [14] the update algorithm adds new (removes existing for deletions) 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. Therefore, the completion graph built prior to the update (e.g., during the initial consistency check) is cached and updated such that if a model exists (i.e, the KB is consistent after the update), a new completion graph will be found. It was observed that updates did not have a large effect on the existing completion graph; therefore, orders of magnitude performance improvements are achieved. This is demonstrated for varying sized datasets (1, 2 and 3 Universities) and updates from the Lehigh University benchmark [11] in Figure 2 (note that incremental consistency checking is denoted by 'Opt', whereas checking consistency from scratch is denoted by 'Reg'). In the figure, it is clear that the performance time of the normal version of the reasoner (three lines at the top of the figure) stays constant through the updates; in contrast the incremental consistency checking approach (the bottom three lines) achieves orders of magnitude performance improvement due to the avoidance of rebuilding the entire comple-

tion graph.

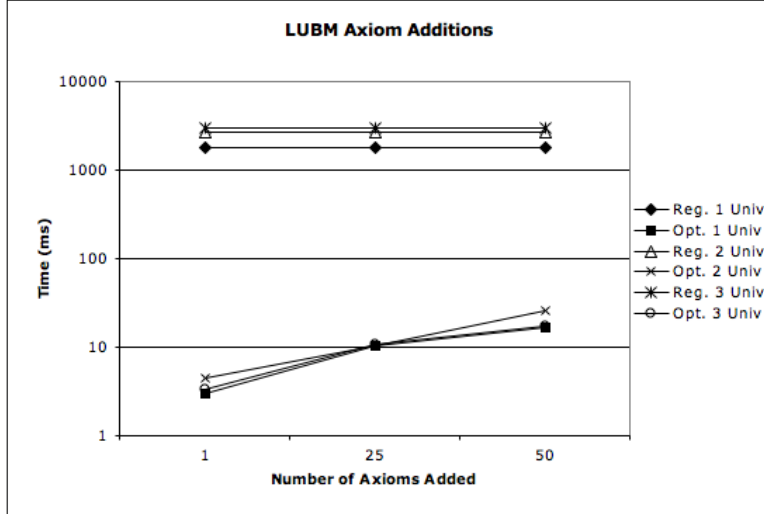


Figure 2: Addition Updates of LUBM datasets

Given this, we adopt this technique for purpose of this work. Further details regarding the approach are omitted here; however they can be found in [14].

4.3 Continuous Query Answering

After guaranteeing consistency of an updated KB, the various subscriptions registered with the broker can be (re)evaluated. In the remainder of this section, we present an approach for more efficient continuous query answering for a subset of OWL-DL, specifically unfoldable *SHI*. Two restrictions are imposed on the queries supported in the approach, namely that only simple roles (i.e., no transitive roles or super-roles of a transitive role) can be used in the query, and the query must contain at least one distinguished variable that the query can be rolled-up into². These restrictions enable the techniques presented in the following sections; further query answering in the presence of transitive roles is a relatively open problem (however, see [10]). In the following sections we assume that all

²Note that more frequently in realistic scenarios, queries contain some number of distinguished variables.

concepts are in negation normal form (i.e., negation only occurs in front of concept names), and all concepts are fully unfolded such that they are composed of only primitive (base) concepts [4].

Before discussing the overall goal of the approach, we make a few simplistic observations. First, due to the monotonicity of *SHI* (and OWL-DL in general) and the (syntactic) additions we consider here, only new answer sets (bindings) can be found. On the other hand, in the event of a (syntactic) deletion from an ABox, by monotonicity we are given that bindings for the query which were not previously entailed by the KB will still not be entailed after the deletion. However, in the case of a deletion a previous binding for the query can possibly be invalidated. Given this, answering continuous queries in the event of ABox additions reduces to determining any new bindings that are entailed by the KB, whereas handling deletions reduces to guaranteeing that previous bindings are still entailed.

4.3.1 Localizing Effects of Updates

When querying very large ABoxes, one of the main problems is that a large number of individuals in the KB must be considered as potential variable bindings. However, we propose that under the types of updates considered in this work, the candidate query bindings can be drastically pruned. A key insight is demonstrated if we consider a simple query such as $\langle x \rangle \leftarrow \text{Company}(x)$. Intuitively, in the event an update is an addition, we would only like to consider affected named individuals not previously bound to x as potential new bindings (i.e., answers); in contrast if the update is a deletion, only individuals previously bound to x that were affected by the update need to be re-checked. Therefore, the main goal of the approach presented here is to localize the named individuals in the KB that are affected by the update in such a way that they may impact the previous query results. 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 12 (*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 clash³ $c = (x, \neg C, C)$ is observed and either $\neg C$ or C is dependent on $l \in \mathcal{L}(y)$ (or node $y \in \mathcal{V}$, edge $\langle y, z \rangle \in \mathcal{E}$, or edge label $R \in \mathcal{L}(\langle y, z \rangle)$), then the clash is said to be dependent on $l \in \mathcal{L}(y)$ (respectively $y \in \mathcal{V}$, $\langle y, z \rangle \in \mathcal{E}$, $R \in \mathcal{L}(\langle y, z \rangle)$). Finally, we say that a label $l \in \mathcal{L}(x)$ is dependent on an update β if β causes the addition of some node label $C \in \mathcal{L}(y)$ (or node $y \in \mathcal{V}$, edge $\langle y, z \rangle \in \mathcal{E}$, or edge label $R \in \mathcal{L}(\langle y, z \rangle)$) s.t. l is dependent on $C \in \mathcal{L}(y)$ (respectively $y \in \mathcal{V}$, $\langle y, z \rangle \in \mathcal{E}$, $R \in \mathcal{L}(\langle y, z \rangle)$); 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 chapter is introduced.

Theorem 1 *Let \mathcal{K} be a \mathcal{SHI} KB, α an ABox addition (resp. deletion), and C some \mathcal{SHI} concept. If $\mathcal{K} \not\models C(a)$ (resp. $\mathcal{K} \models C(a)$) for some $a \in \mathbf{I}_{\mathcal{K}} \cup \mathbf{I}_{\alpha}$, $\mathcal{K} \oplus \alpha \not\models \perp$, and $\mathcal{K} \oplus \alpha \models C(a)$ (resp. $\mathcal{K} \oplus \alpha \not\models C(a)$), then either*

1. *there exists $G \in \text{Comp}(\mathcal{K})$ (resp. $G \in \text{Comp}(\mathcal{K} - \alpha)$) s.t. adding the structures for $\neg C(a)$ and α to G and applying the necessary completion rules results in a clash that is dependent on structures caused by both $\neg C(a)$ and α*
2. *there exists $\{G_1, G_2\} \in \text{Comp}(\mathcal{K})$ s.t. adding α to G_1 and applying the necessary completion rules results in a clash that does not depend on $\neg C(a)$, which closes a branch, D_i , of a disjunction D , and adding $\neg C(a)$ to G_2 and applying the necessary completion rules results in a clash with a label which is dependent on some other branch, D_j ($i \neq j$), of D and the clash does not depend on α*

Proof 1 *See Appendix 9.1 for the proof of this theorem.*

It is important to note that *only if* direction of the theorem does not hold; that is, if some a satisfies the conditions in the theorem, it may not be the case that $\mathcal{K} + \alpha \models C(a)$. However, the theorem does provide a set of necessary conditions for such an entailment. Given this, the remainder of this section presents an approach to determine a superset of the named individuals which could lead to one of the two conditions presented in Theorem 1. To do this, we first introduce various

³Note that due to the fact that number restrictions are not allowed in \mathcal{SHI} , clashes only occur if $\{\neg C, C\} \subseteq \mathcal{L}(x)$ for some node x ; when referring to a clash in the label of node x , the clash will be denoted as a triple $(x, \neg C, C)$.

definitions which intuitively will constitute this superset of named individuals; following this, we show correctness of the approach.

First, we define the notion of *explicitly affected individuals*, which intuitively are the individuals manipulated during the incremental update of all completions for a KB.

Definition 13 (*Explicitly Affected Individuals*): Given *SHI* KB \mathbb{K} and *ABox* update α , define the explicitly affected individuals, denoted $EI(\mathbb{K}, \alpha)$, to be the set of all named individuals $a \in \mathbf{I}_{\mathbb{K}} \cup \mathbf{I}_{\alpha}$ such that either:

1. $a \in \mathbf{I}_{\alpha}$
2. during the update of some $G \in \text{Comp}(\mathbb{K})$ with α (as in Section 4.2), a has some label change, or outgoing/ingoing edge that is added/removed with the following constraints
 - (a) if the update introduces non-deterministic choices, each completion is saturated
 - (b) if a node is reached, then the expansion rules are re-applied to all of the node labels
3. if a clash occurs on node z when updating $G \in \text{Comp}(\mathbb{K})$ then for any node z_i s.t. there exists a path z_0, \dots, z_i ($i \geq 1$) in G where $z = z_0$, z_k a P -neighbor of z_{k-1} ($1 \leq k \leq i$), $P \sqsubseteq R$ and $\{\forall R.C, \forall R^-.C\} \cap \mathcal{L}(z_k) \neq \emptyset$ ($1 \leq k \leq i$), C and P, R some concept and roles respectively, it is the case that either
 - (a) $a = z_i$
 - (b) a is reached by reapply the expansion rules to labels of z_i and any subsequently reached node in some $G \in \text{Comp}(\mathbb{K} \oplus \alpha)$

Additionally, we introduce the notion of a *root path* between two individuals:

Definition 14 (*Root Path*): Define there to be a root path of length D between two nodes x and y of a completion graph if they are reachable by at most D edge traversals where:

1. edge direction is ignored
2. successive traversal of edges labeled with roles that are not simple is only counted once
3. if there exists more than one label for an edge, one of which is not a simple role, then the non-simple edge is traversed and condition 2 is assumed

We now define the general notion of *affected individuals* adopted for the purpose of this work; intuitively, these individuals constitute the named individuals that satisfy Theorem 1 for the rolled-up query concept.

Definition 15 (*Affected Individuals*): Given *SHI* KB \mathbb{K} , conjunctive query Q , and ABox update α , define an individual $a \in \mathbf{I}_{\mathbb{K}} \cup \mathbf{I}_{\alpha}$ to be in the set of affected individuals, denoted $AI(\mathbb{K}, \alpha)$, if either:

1. $a \in EI(\mathbb{K}, \alpha)$
2. for some $b \in \mathcal{V}$ s.t. $D \in \mathcal{L}(b)$, D is of the form $\forall R.C$, and b does not have a P -neighbor ($P \sqsubseteq R$), it is the case that there is a root path of at most length $\text{Depth}(Q)$ between a and b in some $G \in \text{Comp}(\mathbb{K} + \alpha)$ (resp. $G \in \text{Comp}(\mathbb{K})$ for deletions)
3. for some $b \in EI(\mathbb{K}, \alpha)$ there is a root path of at most length $\text{Depth}(Q)$ between a and b in some $G \in \text{Comp}(\mathbb{K} + \alpha)$ (resp. $G \in \text{Comp}(\mathbb{K})$ for deletions).

It can be shown that for there to be a new (resp. invalidated) binding after an update, at least one named individual in the binding must be in $AI(\mathbb{K}, \alpha)$. In order to show this, we first present a variety of properties of models (and completions) for *SHI* KB. First it has been shown in literature that all *SHI* model (and therefore completions) is a forest in which root nodes correspond to named individuals which can have arbitrary edges between them; that is each tree of existential individuals has one unique root which is a named individual [30], which is commonly referred to as the *tree-like model property*. We additionally provide the following lemma regarding role fillers and named individuals, which intuitively states that if there is a R -filler for two named individuals, then it must be the case that either there is a syntactic P -role assertions between the individuals such that $P \sqsubseteq R$ or there is sequence of syntactic P -role assertions between the two individuals. This is a direct consequence of tree-like property of *SHI*, the fact that the tableau expansion rule do not add edges between named individuals, and that updates can only refer to named individuals; thus a formal proof is omitted.

Lemma 1 Given *SHI* KB \mathbb{K} , role $R \in \mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I} \models \mathbb{K}$, complete clash free completion graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \neq)$ corresponding to \mathcal{I} , and $a, b \in \mathbf{I}_{\mathbb{K}}$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ then either

1. R is simple and there exists the syntactic assertion $P(a, b) \in \mathbb{K}$ or $\text{Inv}(P)(a, b) \in \mathbb{K}$ such that $P \sqsubseteq R$

2. there exist a path z_0, \dots, z_k in G with $k \geq 1$, a corresponds to z_0 , b corresponds to z_k , and there exists the syntactic assertion $P(z_i, z_{i-1}) \in \mathbf{K}$ or $\text{Inv}(P)(z_i, z_{i-1}) \in \mathbf{K}$ for $0 \leq i \leq k$ such that $P \sqsubseteq R$

We additionally provide a similar lemma regarding arbitrary role fillers; as this can be shown in the same manner as the proofs for Lemmas A.2 & A.3 in [28], a formal proof is omitted here.

Lemma 2 *Given SHI KB \mathbf{K} , role $R \in \mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I} \models \mathbf{K}$, complete clash free completion graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \neq)$ corresponding to \mathcal{I} , and $a, b \in \Delta^{\mathcal{I}}$, if $(a, b) \in R^{\mathcal{I}}$ then either*

1. R is simple and the node $x \in \mathcal{V}$ corresponding to a must be a R -neighbor of the node $y \in \mathcal{V}$ corresponding to b
2. there exist a path z_0, \dots, z_k in G with $k \geq 1$, a corresponds to z_0 , b corresponds to z_k , and z_i a P -neighbor of z_{i-1} for $0 \leq i \leq k$ and $P \sqsubseteq R$

Additionally, we show that the notion of a root path can be used to determine the propagation of labels in a completion graph due to adding a concept name to some node label.

Lemma 3 *Let \mathbf{K} be a SHI KB, C a SHI concept, and $G \in \text{Comp}(\mathbf{K})$. Also assume C is added to $\mathcal{L}(a)$ for some $a \in \mathbf{I}_{\mathbf{K}}$. If applying the tableau expansion rules to $C \in \mathcal{L}(a)$ and the subsequent labels added as a result of applying the expansion rule to labels added by C causes some label to be added to $\mathcal{L}(b)$, for some $b \in \mathbf{I}_{\mathbf{K}}$, then there exists a root path of at most length $\text{Depth}(C)$ between a and either*

1. b
2. some $c \in \mathcal{V}$ s.t. $\forall R.D \in \mathcal{L}(c)$ for some concept D and role R and c does not have an P -neighbor $d \in \mathcal{V}$ s.t. $P \sqsubseteq R$.

Proof 2 *See Appendix 9.2 for the proof of this lemma.*

Given this we provide the following lemma which provides the necessary conditions for an individual to instantiate (resp. no longer instantiate) a concept after a change in the ABox.

Lemma 4 *Let C be a concept and \mathbb{K} be a SHI KB, and α an ABox addition (resp. deletion). If for some $a \in \mathbf{I}_{\mathbb{K}} \cup \mathbf{I}_{\alpha}$, $\mathbb{K} \not\models C(a)$ and $\mathbb{K} + \alpha \models C(a)$ (resp. $\mathbb{K} \models C(a)$ and $\mathbb{K} - \alpha \not\models C(a)$), then for some $G \in \text{Comp}(\mathbb{K} + \alpha)$ (resp. $G \in \text{Comp}(\mathbb{K})$) one of the following holds:*

1. *for some $b \in \mathcal{V}$ s.t. $D \in \mathcal{L}(b)$, D is of the form $\forall R.C$, and b does not have a P -neighbor ($P \boxtimes R$), it is the case that there is a root path of at most length $\text{Depth}(\neg C)$ between a and b*
2. *for some $b \in \text{EI}(\mathbb{K}, \alpha)$ there is a root path of at most length $\text{Depth}(\neg C)$ between a and b*

Proof 3 *See Appendix 9.3 for the proof of this lemma.*

It can be observed that the first condition of Lemma 4 can cause a substantial number of individuals to be considered as candidates for instantiating (resp. no longer instantiating) some concept after an update. However, we point out this case can be avoided if either 1) the KB does not contain inverses 2) the concept obtained when $\neg C$ is fully unfolded does not contain a concept of the form $\exists R.D$ or 3) for each sub-concept of the form $\exists R.D$ obtained when unfolding $\neg C$, it is the case that $\text{Inv}(P) \notin \mathbb{K} + \{\neg C(a)\} + \alpha$ where $R \boxtimes \text{Inv}(P)$. This is because concept labels will not be able to be transferred back up a new edge added as a result of adding $\neg C \in \mathcal{L}(a)$ to any completion. First, we provide a lemma similar to Lemma 3, which takes into account this restriction.

Lemma 5 *Let \mathbb{K} be a SHI KB, C a SHI concept, $G \in \text{Comp}(\mathbb{K})$, and that one of the following holds:*

1. *C and \mathbb{K} do not contain inverses*
2. *the concept obtained when $\neg C$ is fully unfolded does not contain a sub-concept of the form $\exists R.D$*
3. *for each sub-concept of the form $\exists R.D$ obtained when unfolding $\neg C$, it is the case that $\text{Inv}(P) \notin \mathbb{K} \cup \{\neg C(a)\}$ where $R \boxtimes \text{Inv}(P)$*

Also assume C is added to $\mathcal{L}(a)$ for some $a \in \mathbf{I}_{\mathbb{K}}$. If applying the tableau expansion rules to $C \in \mathcal{L}(a)$ and the subsequent labels added as a result of applying the expansion rule to labels added by C causes some label to be added to $\mathcal{L}(b)$, for some $b \in \mathbf{I}_{\mathbb{K}}$, then there exists a root path of at most length $\text{Depth}(C)$ between a and b .

Proof 4 See Appendix 9.4 for the proof of this lemma.

Given this, the conditions in Lemma 4 can be relaxed if the restriction holds; a proof is omitted as given Lemma 5 this can be in the same manner as the proof for Lemma 4.

Lemma 6 Let C be a concept, α an ABox addition (resp. deletion) and \mathbb{K} be a SHI KB. Additionally, assume that one of the following holds:

1. C , α and \mathbb{K} do not contain inverses
2. the concept obtained when $\neg C$ is fully unfolded does not contain a sub-concept of the form $\exists R.D$
3. for each sub-concept of the form $\exists R.D$ obtained when unfolding $\neg C$, it is the case that $\text{Inv}(P) \notin \mathbb{K} \cup \{\neg C(a)\} \cup \alpha$ where $R \sqsubseteq \text{Inv}(P)$

If for some $a \in \mathbf{I}_{\mathbb{K}} \cup \mathbf{I}_{\alpha}$, $\mathbb{K} \not\models C(a)$ and $\mathbb{K} + \alpha \models C(a)$ (resp. $\mathbb{K} \models C(a)$ and $\mathbb{K} - \alpha \not\models C(a)$), then for some $G \in \text{Comp}(\mathbb{K} + \alpha)$ (resp. $G \in \text{Comp}(\mathbb{K})$) there exists $b \in \text{EI}(\mathbb{K}, \alpha)$ s.t. there is a root path of at most length $\text{Depth}(\neg C)$ between a and b

Lemma 6 implies that in the event that the KB satisfies these constraints, then Definition 15 can be relaxed; specifically, the second condition can be ignored, making the approach potentially even more effective. Another very important consequence of Lemma 6 is that range constraints can be used if the KB satisfies the restrictions imposed. Unfortunately, domain and range constraints violate the restriction that the KB be unfoldable; for example, asserting that the range of a role R is the concept D is equivalent to asserting $\top \sqsubseteq \forall R.D$, which is clearly not unfoldable (note $\top \equiv C \sqcup \neg C$), and in fact, is a general concept inclusion (GCI) axiom. However, it has previously been shown that domain and range constraints can be handled by additional tableau expansion rules [31], rather than treating them as normal GCIs; further, these additional tableau expansion rules are supported in axiom tracing [20] and the incremental consistency checking approach [14]. Given this, it can easily be shown that allowing range constraints (under the assumption that the KB satisfies the restriction in Lemma 6) does not affect the approach previously described; this is because Lemma 3 can easily be shown to still hold when $b \in \mathbf{I}_{\mathbb{K}} \cup \mathbf{I}_{\alpha}$, and the clashes that range constraints contribute to when $\neg C(a)$ is added to any completion will always be on non-root nodes. Unfortunately, the same cannot be shown for domain constraints and this is left as

future work. However, we do note it can easily be shown that if the concept C has been fully unfolded and there does not exist a sub-concept of the form $\exists R.D$ (i.e., condition 2 of Lemma 6), then neither domain or range constraints affect the approach and can therefore be allowed.

Now we provide the following proposition showing that for there to be a new (resp. invalidated) binding then at least one named individual in that binding will be in the set of affected individuals.

Proposition 1 *Let K be a SHI KB, Q a conjunctive query, and α an ABox update. If $K \not\models Q[x_1/a_1, \dots, x_n/a_n]$ and $K \oplus \alpha \models Q[x_1/a_1, \dots, x_n/a_n]$ (resp. $K \models Q[x_1/a_1, \dots, x_n/a_n]$ and $K \oplus \alpha \not\models Q[x_1/a_1, \dots, x_n/a_n]$), then there exists some named individual $b \in \mathbf{I}_K \cup \mathbf{I}_\alpha$ that is bound to some $y \in V(Q)$ such that $b \in AI(K, \alpha)$.*

Proof. See Appendix 9.5 for the proof of this proposition. \square

Proposition 1 is intuitive as it states that for there to be a new (resp. invalidated) binding, then there must exist some individual in that binding that either is directly affected by the update in some completion graph or is in the *proximity* of some other individual that was directly affected. We are able to show that the notion of proximity introduced in Proposition 1 (conditions 2 and 3) is sufficient (observe that currently we do not take into account the structure of the concepts in the query; however, we plan to address this in future work). More importantly, Proposition 1 implies that in order to find the affected individuals, one can update all $G \in Comp(K)$ and gather the individuals that satisfy the properties provided; note that individuals satisfying the second/third property can be found using a simple depth first search of length D (with the restrictions provided in Definition 14).

It is clear, however, that 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, saturating the tableau is a very expensive process. To avoid performing a full saturation of the initial KB, we propose building a structure that we refer to as a *summary root graph*.

Definition 16 (*Summary Root Graph*): *Let G be the completion graph built for SHI KB K by applying all tableau expansion rules to K as normal, however with the following modifications:*

1. *if a non-deterministic choice is encountered (i.e., a disjunction), add all labels in the disjunction to the node and proceed without creating a new branch*

2. if a clash is encountered, it is ignored

Define the summary root graph S_G as $S_G = \text{Roots}(G)$

Observe that condition 2 is required, as adding all labels from a disjunction can obviously introduce clashes; also note that only the structure for root nodes and their edges is kept to reduce memory overhead as the discarded structures can be rebuilt when necessary using the existing root node labels. It can be seen that the summary root graph does not correspond to a model of the KB; rather, a overestimate of all root node and edge labels that would be obtained by saturating the original KB. This approach guarantees that if a root node (corresponding to a named individual) or edge between root nodes has a label in some completion graph corresponding to a model for the KB, then that label will be in the label set for that individual in the summary root graph. The aim behind the approach is to use this structure to localize an overestimate of the explicitly affected individuals; an overestimate is acceptable as, in the end, we are trying to find only *candidate* distinguished variable bindings. Further, the approach is substantially more efficient than a normal saturation, as branches are not created during the modified application of expansion rules.

Summary Root Graph Properties. In this section, various properties of the summary root graph for a KB are shown. First it is observed that termination of the construction of the summary root graph trivially follows as termination for the *SHI* tableau algorithm is independent on clash detection. Next, we show that if a root node has a label in some completion graph corresponding to a model for the KB, then that concept name will be in the label for that individual in the summary root graph. Further, properties regarding the tree and graph-like structures rooted at root nodes are shown. These properties will be utilized later when showing the correctness of the approach.

First, the following notation is introduced: given a completion graph G with root node x , denote by $\text{Tree}(x)$ the tree rooted at x that is composed of x , all non-root descendants of x , and the labels for nodes and edges for the tree; this tree is referred to as the *root tree* for x in G . Additionally, given root tree T , denote by $\text{Root}(T)$, V_T , E_T , and L_T the unique root, set of nodes, edges, and label function for the tree respectively. The notion of sub-graphs of root nodes in a completion graph is also introduced. This structure can be viewed as a generalization of a root tree in which node neighbors, edges from the predecessor of a blocked node to the blocking node, and edges between root nodes are also considered.

Definition 17 (Root Graph) A root graph G is composed of a set of nodes V_G , edges E_G , and labeling function L_G for the nodes and edges; additionally, there is a uniquely defined root node of the graph, $\text{Root}(G)$. Given a completion graph G and root node x , the root graph G for x , denoted $\text{Graph}(x)$, is defined as follows:

1. $\text{Root}(G) = x$, $x \in V_G$, and $L_G(x) = \mathcal{L}(x)$
2. if $y \in V_G$ and y has R -neighbor z in G s.t. z not blocked, then $z \in V_G$, $\langle y, z \rangle \in E_G$, $L_G(z) = \mathcal{L}(z)$ and $L_G(\langle y, z \rangle) = L_G(\langle y, z \rangle) \cup \{R\}$
3. if $w, y \in V_G$, $z \in \mathcal{V}$ a non-root node, y the predecessor of z , y has R -neighbor z , and w blocks z in G , then $\langle y, w \rangle \in E_G$ and $L_G(\langle y, w \rangle) = L_G(\langle y, w \rangle) \cup \{R\}$

Next, the notion of *tree containment* in a root graph is introduced; intuitively, this implies that the tree structure of the root node is included in the graph structure of the node.

Definition 18 (Tree Containment) Let T, G be a root tree and root graph respectively. Inductively define node $x \in V_T$ to be contained in $y \in V_G$, denoted $\text{con}(x, y)$, if the following holds:

1. $L_T(x) \subseteq L_G(y)$
2. for each $\langle x, z \rangle \in E_T$ there exists an edge $\langle y, w \rangle \in E_G$ s.t.
 - (a) for all $R \in L_T(\langle x, z \rangle)$ there is some $S \in L_G(\langle y, w \rangle)$ s.t. $S \sqsubseteq R$ and
 - (b) $L_T(z) \subseteq L_G(w)$ and
 - (c) $\text{con}(z, w)$

Then, T is said to be contained in G , denoted $\text{contain}(T, G)$, if $\text{con}(\text{Root}(T), \text{Root}(G))$. Lastly, given a root tree T and root graph G s.t. $\text{contain}(T, G)$, denote by $y \rightarrow_{T,G} z$ a mapping of node $y \in V_T$ into $z \in V_G$ s.t. the containment relationship is satisfied.

Importantly, each root tree in a complete and clash-free completion graph for K must be contained in the root graph for the corresponding node in the summary root graph for K prior to removing all trees rooted at root nodes.

Lemma 7 Let K be a SHI KB and S_G be the summary root graph for K obtained prior to removing all trees rooted at root nodes. Then for all $G \in \text{Comp}(K)$ and each $a \in \mathbf{I}_K$, $x_a \in \mathcal{V}$ and $x'_a \in \mathcal{V}_{S_G}$, it is the case that $\text{contain}(\text{Tree}(x_a), \text{Graph}(x'_a))$.

Proof. See Appendix 9.6 for the proof of this lemma. \square

Because root nodes are never blocked and the tableau expansion rules do not add edges or edge labels between root nodes, it must be the case that all edges and their labels between root nodes that exist in any complete and clash free completion graph for the KB also exist in the summary root graph. Intuitively, this and Lemma 7 imply that the summary completion subsumes the information in all complete and clash-free completion graphs for K.

Using the Summary Root Graph. Using the summary root graph, an overestimate for $EI(K, a)$ can be provided; we first note that in the case of ABox *deletions*, we propose using axiom tracing [3, 20, 14] during the application of expansion rules, effectively tracking the asserted axioms responsible for changes to the summary root graph. The components of the graph that are dependent on the deleted axiom can therefore be identified using the axiom tracing function defined in [14]. We also note that there is a simple modification when checking if the expansion rules can be applied to a node (to guarantee completeness). Specifically, node labels are marked when they have had a completion rule applied to them during the overestimate procedure; if the label is not marked, then the expansion rule is applied. The modified expansion rules are provided in Table 2.

\sqcap -rule:	if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not directly blocked and either a) $C_1 \sqcap C_2$ not marked or b) $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then mark $C_1 \sqcap C_2$ and $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
\sqcup -rule:	if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not directly blocked and either a) $C_1 \sqcup C_2$ not marked or b) $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then mark $C_1 \sqcup C_2$ and $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
\exists -rule:	if $\exists S.C \in \mathcal{L}(x)$, x is not directly blocked and either a) x has no S -neighbor y with $\{C \in \mathcal{L}(y)$ or b) $\exists S.C$ not marked then mark $\exists S.C$ and create a new node y with $\mathcal{L}(\langle x, y \rangle) = S$ and $\mathcal{L}(y) = C$
\forall -rule:	if $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked and either a) there is an S -neighbor y of x with $C \notin \mathcal{L}(y)$ or b) $\forall S.C$ is not marked and there is an S -neighbor y of x with $C \in \mathcal{L}(y)$ then mark $\forall S.C$ and $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$
\forall_+ -rule:	if $\forall S.C \in \mathcal{L}(x)$, there is some R with $Trans(R)$ and $R \boxplus S$, x is not indirectly blocked and either a) there is an R -neighbor y of x with $\forall R.C \notin \mathcal{L}(y)$ or b) $\forall R.C$ is not marked and there is an R -neighbor y of x with $\forall R.C \in \mathcal{L}(y)$ then mark $\forall R.C$ and $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\forall R.C\}$

Table 2: Modified *SHI* tableau expansion rules used in Definition 13

It can be shown that the approach for updating the summary root graph terminates and that after the update Lemma 7 still holds.

Lemma 8 *Let \mathcal{K} be a SHI KB, $G \in \text{Comp}(\mathcal{K})$, S_G the summary root graph for \mathcal{K} , α an ABox addition, G' the result of adding α to G (either containing a clash, or complete and clash-free), and S'_G be the result of updating S_G with α . Then updating S_G terminates and Lemma 7 holds for S'_G and G' .*

Proof. See Appendix 9.7 for the proof of this lemma. \square

In order to show completeness of the approach, we must demonstrate that S_G can be used to find all individuals that satisfy the conditions of Definition 13. Clearly, the first condition is trivial. Consider the second condition for addition updates. 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. This, in conjunction with the marking scheme, implies that it suffices to show that if a label is added to some root node x_a during the update of some $G \in \text{Comp}(\mathcal{K})$, then a label will be (re)added to the corresponding root node in the summary root graph.

Lemma 9 *Let \mathcal{K} be a SHI KB, $G \in \text{Comp}(\mathcal{K})$, S_G be the summary root graph for \mathcal{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 .*

Proof. See Appendix 9.8 for the proof of this lemma. \square

Now, let us consider the third condition of Definition 13 for addition updates. It can be shown that all clashes observed when updating each $G \in \text{Comp}(\mathcal{K})$ will be observed when updating the summary root graph. While clashes are ignored during the construction and update of the summary root graph, they still will be present and therefore can be tracked.

Lemma 10 *Let \mathcal{K} be a SHI KB, $G \in \text{Comp}(\mathcal{K})$, S_G be the summary root graph for \mathcal{K} , and β a set of ABox assertions. Also assume that there is a clash $c = (y, C, \neg C)$ observed when β is added to G and let x be the unique root of y (possibly y itself). Additionally, let $T = \text{Tree}(x)$ in G at the time of the clash and $G =$*

Graph(x) in S_G after S_G is updated with β . Then, for all $y \rightarrow_{T,G} z$, the expansion rules will be applied to z when updating S_G and $\{\neg C, C\} \subseteq \mathcal{L}_{S_G}(z)$.

Proof. See Appendix 9.9 for the proof of this lemma. \square

This in conjunction with the tree containment property of the summary root graph (shown in Lemmas 7 & 8) implies that the third condition of Definition 13 can easily be accounted for by simply using the summary root graph. Thus, we have effectively shown the summary root graph can be used to the explicitly affected individuals under an addition update.

Importantly, due to the completeness of axiom tracing, which is assumed to be used when constructing and maintaining the summary root graph, deletion updates can easily be supported in a similar manner. Given this, we now formally define the overestimate of explicitly affected individuals found using the summary root graph.

Definition 19 (*Overestimate of Explicitly Affected Individuals*): *Let S_G be the summary root graph for \mathcal{SHI} KB \mathbb{K} and α an ABox update. Define the overestimate of explicitly affected individuals, denoted $EI_{S_G}(\mathbb{K}, \alpha)$, to be defined by the following procedure:*

1. (a) α an addition: *add the structure introduced by the update to S_G (as in [14]) and apply the tableau expansion rules to all labels of individuals x of S_G such that $x \in \mathbf{I}_\alpha$*
 (b) α a deletion: *remove all structures from S_G dependent on the deleted assertion (determined as in [14]) and apply the expansion rules to all individuals and their neighbors if the node was affected by the initial retraction of structures dependent on α*
2. *apply the expansion rules to all labels of individuals (named or un-named) reached by subsequent rule firings*
3. *use the modifications to the tableau expansion rules in Definition 16 and Table 2*
4. *if a clash is found, then condition 3 of Definition 13 is checked however, the S_G is used rather than all $G \in \text{Comp}(\mathbb{K})$.*

$EI_{S_G}(\mathbb{K}, \alpha)$ is then composed of all root nodes that are reached during the application of expansion rules, have a label change, or are adjacent to an edge or edge label that changed.

We point out here that it is critical to apply completion rules to all labels of a node when it is reached during the application of expansion rules in order to guarantee the completeness of the approach (i.e., the structures discarded from the initial summary graph corresponding to un-named individuals must be rebuilt). Observe that this is an overestimate, as the summary root graph is itself an overestimate of node labels and clashes are ignored. Lastly, note that after the application of expansion rules finishes, it is assumed that un-named nodes and their edges are discarded from the summary root graph (which has therefore been updated). Note that this technique proceeds in a similar way as our previous work on incremental consistency checking [14], with the exception of the approach used when applying the expansion rules. Additionally, when the summary root graph is updated during an addition, axioms traces are updated using the same approach as in [14]. Completeness of the overestimate of the explicitly affected individuals follows from the previous assumptions and Lemmas 7–10.

Proposition 2 *Given a SHI KB \mathcal{K} , ABox update α and summary root graph S_G for \mathcal{K} , then the approach for finding $EI_{S_G}(\mathcal{K}, \alpha)$ is terminating and $EI(\mathcal{K}, \alpha) \subseteq EI_{S_G}(\mathcal{K}, \alpha)$.*

Proposition 2 and Lemmas 7 & 8 imply that we can use S_G to locate a superset of the affected individuals (defined below). We importantly note that it is obvious that some node in S_G can have a label of the form $\forall R.D$ and have a R -neighbor which may not exist in some $G \in \text{Comp}(\mathcal{K} + \alpha)$. This implies that condition 2 in the approach for finding the affected individuals (i.e., Definition 15) can potentially be problematic when using S_G rather than all $G \in \text{Comp}(\mathcal{K} + \alpha)$. However, note that this is trivially overcome as axiom tracing is assumed when building S_G ; therefore one can simply consider any node in S_G s.t it has a label of the form $\forall R.D$ and either does not have an R -neighbor or has an R -neighbor that is dependent on some disjunction (clearly an overestimate).

Definition 20 (*Overestimate of Affected Individuals*): *Let \mathcal{K} be a SHI KB, Q a conjunctive query, S_G the summary root graph for \mathcal{K} , α an ABox update and $AI_P(\mathcal{K}, \alpha)$ the set of individuals satisfying conditions 2 or 3 from Definition 15 s.t. $b \in EI_{S_G}(\mathcal{K}, \alpha)$ is used rather than $b \in EI(\mathcal{K}, \alpha)$. Define the overestimate of affected individuals, denoted $AI_O(\mathcal{K}, \alpha)$, as $AI_O(\mathcal{K}, \alpha) = EI_{S_G}(\mathcal{K}, \alpha) \cup AI_P(\mathcal{K}, \alpha)$.*

Discussion. It is clear that there are possible limitations to the current approach for determining the overestimate of individuals affected by updates. In particular, if the approach produces an overestimate that is too large, the value of

the approach may degrade (note that in the worse case, the number of individuals one would have to check is the same as in the non-incremental case). However, our initial results indicate that the approach is extremely effective. An additional limitation of the approach is the memory overhead imposed by maintaining the summary root graph, namely maintaining the node and edge labels; this is clearly a trade-off in the approach. One last issue is related to the application of expansion rules on the summary root graph *with respect to the update*. We point out here that in the worst case this could impose overhead that is not practical for the performance demands of some syndication applications; however, our initial results demonstrate that this is not the case. This is because the expansion rules are applied with respect to only the update and not the entire KB. This is actually quite intuitive, as one would expect for updates to only affect a small portion of the KB. Further, if we consider syndication of news feeds for example, one would expect updates to be generally focused on a small number of individuals. This is clearly evident in the financial domain; for example, the Dow Jones Newswires disseminates on average 10,000 news feeds per day⁴, which are typically terse and focused on specific companies, industries, etc.

4.3.2 Query Impact on Candidate Individuals

When a query contains roles and more complex query patterns, considering only directly affected individuals as potential new bindings (resp. invalidated bindings for deletions) will not suffice. For example, consider the following query for all companies that have sell recommendations: $\langle x, y \rangle \leftarrow \text{onRecommendation}(x, y) \wedge \text{Company}(x) \wedge \text{SellList}(y)$. Also assume that there is an ABox addition that Ford is a *Company* and that after the update, $AI_O(\mathbf{K}, \alpha)$ only includes Ford. It is clear we cannot simply consider Ford as the only candidate binding for the variables in the query, as there could exist any number of individuals (i.e., instances of *SellList*) related to Ford by an *onRecommendation* role. Therefore, it can be observed that the query structure/shape impacts the affected individuals that must be considered as bindings for distinguished variables; we refer to this as the *query impact*.

We now introduce a technique for determining the query impact on the affected individuals, which is a straightforward approach that induces very little overhead. The key insight is that for a new query binding to be entailed (or invalidated in the case of deletions) under syntactic ABox updates in *SHI*, at least one named individual that is bound to some $x \in V(Q)$ must be in $AI_O(\mathbf{K}, \alpha)$ (by Proposi-

⁴Source: <http://www.djnewswires.com/us/djtotalcoverageinfo.htm>

tion 1). This, along with the facts that the query is assumed to be connected and only simple roles are used in queries, implies that given an addition update, the query impact on the original affected individuals can be taken into account by also considering all named individuals in the updated completion graph (discussed in Section 4.2) that are reachable from some $x \in AI_O(\mathbb{K}, \alpha)$ by at most n edge traversals (with the direction ignored), where n is the longest path in the query graph; this can easily be accomplished by a breadth first search of maximal depth n starting at all $x \in AI_O(\mathbb{K}, \alpha)$. Given this, under additions only the various combinations of individuals in this expanded set of affected individuals need to be considered as possible new bindings for distinguished variables. A similar approach can be used to take into account the query impact under deletions, however the original completion graph (prior to its update) must be used for the search of depth n (i.e., deletions can remove structures from the completion). In the case of deletions, one then needs to recheck any previous binding that contains some individual in the expanded set of affected individuals. Denote by $query_impact(AI_O(\mathbb{K}, \alpha), Q)$ the extended set of affected individuals under this approach for taking into account the query impact.

Proposition 3 *Let \mathbb{K} a \mathcal{SHI} KB, Q a conjunctive query, and α be an ABox update. Then $query_impact(AI_O(\mathbb{K}, \alpha), Q)$ is terminating and under a syntactic update, in order for a new (resp. invalidated) variable binding, $Q[x_1/a_1, \dots, x_n/a_n]$, to be entailed (resp. no longer entailed) after an addition (resp. deletion) α , then $\{a_1, \dots, a_n\} \in query_impact(AI_O(\mathbb{K}, \alpha), Q)$.*

Proof. See Appendix 9.10 for the proof of this proposition. \square

It is clear that the previous technique does not leverage the actual structure of the query (i.e., concepts and roles in the query); therefore, we now introduce a more effective approach that exploits such information, but also introduces additional overhead. In this discussion, the approach is only presented for additions, as in typical syndication systems, updates are much more frequently additions; further, extending the approach to deletions is straightforward. We first point out that it has previously been shown that a conjunctive query can be answered by syntactically *mapping* the query into all completion graphs for the KB [23, 27]. More specifically for the DL \mathcal{SHIQ} (also applicable to \mathcal{SHI}), [27] shows that given a completion graph G and a query Q , the query can be mapped into G , denoted $Q \hookrightarrow G$, if there is a mapping μ from the variables (both distinguished and non-distinguished) and individuals in the Q into the nodes of G such that

- $\mu(a) = a$ for each individual a ,
- 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 completions, then the KB satisfies the query. It can be seen that such a mapping is usable when of taking into account the query impact under additions. We note, however, that such a mapping introduces overhead, as it requires that the KB must be extended with $\top \sqsubseteq C \sqcup \neg C$ for each concept $C \in \text{Con}(Q)$. In order to further reduce the new candidate bindings, each individual in $AI_o(\mathbb{K}, \alpha)$ can be iteratively substituted into variables in the query; neighbor nodes in the updated completion graph can then be inspected to see if they can be mapped into the remaining nodes (via roles whose labels match the query graph) in the query graph (note that distinguished variables in the query graph are mapped into nodes corresponding to named individuals). If there does not exist a mapping in which a given named individual can be mapped into a particular distinguished variable, then this individual does not need to be considered in the candidate distinguished variable set for this variable; this is because we have just found a completion graph (i.e., model) in which the query cannot be mapped [27]. However, if a named individual can be mapped into a distinguished variable, then we must consider this individual as a candidate binding.

For example, if we consider the query $\langle x,z \rangle \leftarrow \text{Company}(x) \wedge \text{onSellList}(x,y) \wedge \text{hasCEO}(y,z)$, the following query graph is obtained (note that the variable y is non-distinguished): When we look at the query impact on the affected set, any



Figure 3: Sample Query Graph

affected individual that does not have a *onSellList* successor, which also has a *hasCEO* successor cannot be a candidate binding for the variable x . We now formally define the set of named individuals that must be considered as candidates for distinguished variables under *Query Impact*.

Definition 21 (*Query Impact on Candidate Bindings*): Let \mathbb{K} be a SHI KB, Q a conjunctive query, α an ABox addition and $G \in \text{Comp}(\mathbb{K} \oplus \alpha)$. Define the set of candidate bindings for distinguished variable x under query impact, denoted $A_{QI}(x)$, as follows:

$$A_{Q_I}(x) = \{a \mid a \in AI_O(K, \alpha) \wedge Q \hookrightarrow_{\{x \leftarrow a\}} G\} \cup \\ \{a \mid b \in AI_O(K, \alpha) \wedge Q \hookrightarrow_{\{x \leftarrow a, y \leftarrow b\}} G\}$$

where $Q \hookrightarrow_{\{x \leftarrow a\}} G$ denotes a mapping of Q into G with the restriction that the distinguished variable x must be mapped into the named individual a in the completion graph.

Proposition 4 *Query Impact on candidate bindings, denoted $A_{Q_I}(x)$, is terminating and complete (all candidate variable bindings will be found) and terminating.*

Proof. See Appendix 9.11 for the proof of this proposition. \square

4.3.3 Continuous Query Answering Algorithm

We now describe the algorithm for answering continuous ABox queries. Similar to the discussion presented above, the algorithm is presented in terms of a single query. Note that the algorithm utilizes a combination of both techniques for taking into account *query impact*. It is assumed the KB is first preprocessed such that for each $C \in Con(Q_c)$, an axiom $\top \sqsubseteq C \sqcup \neg C$ is added to the KB; this is necessary only in the cached completion graph for consistency checking and not in the summary root graph. Additionally, it is assumed the summary root graph is created at startup and that the initial set of answers for Q_c is previously determined.

Algorithm 1 presents the function *SubReg* which registers a subscription (i.e., continuous query) and first performs the initial query over the KB. It is assumed that standard techniques for rolling up the query are used to checked for entailment (e.g., see [19]). This generates the initial answer set for the query that will be continuously maintained as the ABox is updated through subsequent publications. The method *Update* is the ABox update and incremental consistency checking method as defined in [14].

Algorithm 2 presents the main continuous query answering algorithm. The approach first locates the affected individuals; this is denoted by *localize_effects*(S_G, α) and takes as input an initial summary root graph S_G and update α and is assumed to both update S_G and return the affected individuals. Additionally, the extended set of affected individuals is found using the simple query impact. If the update is an addition, the set of candidate distinguished variable bindings (determined using Definition 21) is iterated over and checked for entailment. It is assumed that standard techniques for query answering are used (e.g., see [19]). If the update is a deletion, each tuple in the previous answer set is iterated over. Previous tuples

that do not contain some individual in $query_impact(AI_O(K, \alpha), Q)$ are still valid, as the update did not affect any of the bound individuals; otherwise, the tuples are re-checked for entailment.

Proposition 5 *Algorithm 2 is sound, complete, and terminating.*

Proof. See Appendix 9.12 for the proof of this proposition. \square

Algorithm 1 $sub_reg(K, Q_c, B)$

Input:

K: initial KB
 Q_c : continuous query
 B : update buffer

1: $R \leftarrow answer_query(Q, K)$
2: $S_G \leftarrow get_summary_root_graph(K)$
3: **while** Q_c not expired **do**
4: $\alpha \leftarrow$ next update in B
5: $K, R, S_G \leftarrow update_query_results(K, S_G, Q_c, R, \alpha)$
6: **end while**

5 Empirical Results

In order to evaluate the continuous query answering approach presented in this paper for the purpose of syndication, we have implemented the basic functionality of the framework defined in Section 3 and the algorithm presented in Section 4. In the current implementation, publishers can register and publish information (currently all information remains indefinitely valid). Additionally, subscribers can register subscriptions in the form of continuous conjunctive queries that can remain valid for varying amounts of time. In the evaluation, we have focused on information matches as the technical contributions of this work mainly address scalability issues with respect to this problem. Further evaluation of publication matches is left as future work, however initial insights are provided in Section 6.

We have performed an empirical evaluation using the Lehigh University Benchmark (LUBM) [11] (*SHI* expressivity), as it provides a large ABox, therefore simulating a broker with large numbers of persistent publications; additionally, it supplies queries with similar complexity as those used in the examples throughout this paper. It should be noted that 8 OWL equivalent class axioms were changed

Algorithm 2 *update_query_results*(K, S_G, Q_c, R, α)

Input:

K : initial KB
 S_G : summary root graph for K
 Q_c : continuous conjunctive query
 R : set of all current bindings (answer set)
 α : ABox update

Output:

K : updated KB
 S_G : updated summary root graph
 R : updated bindings (answer set)

```
1:  $K \leftarrow K \oplus \alpha$ 
2: if  $K$  is not consistent then
3:    $K \leftarrow$  Retract  $\alpha$  from  $K$ 
4:   return  $K, S_G, R$ 
5: end if
6:  $AI_O(K, \alpha) \leftarrow localize\_effects(S_G, \alpha)$ 
7:  $QI_S \leftarrow query\_impact(AI_O(K, \alpha), Q)$ 
8: if  $\alpha$  is an addition then
9:   for all  $a_1 \in A_{QI}(x_1), \dots, a_n \in A_{QI}(x_n)$  s.t.  $x \in DV(Q_c)$  do
10:    if  $K \models Q_c[x_1/a_1, \dots, x_n/a_n]$  then
11:       $R \leftarrow R \cup \{ \langle a_1, \dots, a_n \rangle \}$ 
12:    end if
13:  end for
14: else if  $\alpha$  is a deletion then
15:   for all  $\langle a_1, \dots, a_n \rangle \in R$  do
16:    if  $QI_S \cap \{a_1, \dots, a_n\} = \emptyset$  then
17:      continue
18:    else if  $K \not\models Q_c[x_1/a_1, \dots, x_n/a_n]$  then
19:       $R \leftarrow R \setminus \{ \langle a_1, \dots, a_n \rangle \}$ 
20:    end if
21:  end for
22: end if
23: return  $K, S_G, R$ 
```

to subclass axioms, so that the KB was unfoldable. Three queries from LUBM were selected as sample subscriptions and continuously run over a dataset comprised of one university, containing 16,283 individuals and 78,094 assertions. The following three queries were used in the evaluation (LUBM queries 1, 3, and 13 respectively):

```
 $\langle x \rangle \leftarrow \text{GraduateStudent}(x) \wedge \text{takesCourse}(x, \text{GraduateCourse} \mathbf{0})$   
 $\langle x \rangle \leftarrow \text{Publication}(x) \wedge \text{publicationAuthor}(x, \text{AssistantProfessor} \mathbf{0})$   
 $\langle x \rangle \leftarrow \text{Person}(x) \wedge \text{hasAlumnus}(\text{University} \mathbf{0}, x)$ 
```

In the evaluation, the queries were run over the KB, which was updated with a collection of ABox assertions, simulating newly published information; updates were randomly selected individual type (atomic) and/or role assertions, as this aligns with the types of updates one would expect in syndication systems. Each published document was indefinitely valid. To ensure that some of the updates affected the query results (i.e., subscriptions), there was a 50-percent probability that the update referred to one of the individuals bound to a distinguished variable; therefore, approximately half of the selected updates actually affected the subscriptions. Tests were performed for each update type (additions and deletions) using varying update sizes; namely 1, 5, 10, 15, and 25 assertions. This therefore provided insights into the effectiveness of the technique with respect to the size of the published documents. Each test was performed 25 times, and the results were averaged. Lastly, the experiments were performed using a 1.5 GHz processor with 1GB of memory.

In the evaluations, two versions of the DL reasoner Pellet⁵ were used; a regular version of the reasoner and an optimized reasoner that used the techniques presented in this paper. In the regular version of the reasoner, the standard query answering algorithm was performed after each update. Additionally, the KAON2⁶ OWL-DL reasoner was used in the evaluation. KAON2 reduces OWL KBs to disjunctive datalog and is optimized for query answering. Additionally, KAON2 was used as it provides functionality to add and remove assertions and re-run queries after a KB has been updated (we do note that it is unclear whether KAON2 currently performs view maintenance). Using this as a comparison, we aimed to provide interesting insights into tableau-based algorithms for syndication purposes when compared to other possible approaches.

⁵Pellet project homepage: <http://www.mindswap.org/2003/pellet/>

⁶KAON2 project homepage: <http://kaon2.semanticweb.org/>

Results for continuous query answering for the various LUBM queries are presented in Figure 4; both addition and removal results are presented for the regular version of Pellet, KAON2, and the optimized version of Pellet (denoted “Pellet-C” in the figure). All timing results are shown in milliseconds and the scale is logarithmic. Note that the “0” update size value represents the time to run the initial query prior to performing an update (this includes the start-up cost for the continuous query answering approach). In all three queries, the initial query answering times (prior to any update) in both the regular version of Pellet and the optimized version were comparable. This is because the generation of the summary root graph introduced little overhead; specifically, the average time to build the summary root graph was only 2.7 seconds (on average 500 milliseconds larger than the initial consistency check). We note that there is little overhead because of the small amount non-determinism in LUBM (primarily due to making the KB unfoldable). For exposition, however, we investigated the time to build the summary root graph for the original version of LUBM, which induce a large amount of non-determinism. It was observed that the average time to build the summary root graph took approximately 26.1 seconds. This demonstrates the expected impact of a substantial amount of non-determinism on the approach; while this introduces overhead, we argue that this is acceptable, as it is only performed once at startup. Further, the generation of the summary root graph was far more efficient than the alternative of saturating the initial KB, as this would not terminate.

For both update types, approximately one to three orders of magnitude performance improvements are achieved over the regular version of Pellet by using the optimized matching algorithm as updates are received. This is due to the incremental consistency checking approach and the reduction of candidate variable bindings. In the evaluation we observed that in all queries, the average incremental consistency checking time was approximately 7 milliseconds, where the normal consistency checking time in Pellet was approximately 2,200 milliseconds. This illustrates the utility of the incremental consistency checking approach for the purpose of syndication. Additionally, in the three queries the actual query answering time (excluding consistency checking) for the regular version of Pellet was on average between 500 to 1,000 milliseconds. In contrast, using the optimizations presented in this work, the average query answering time (excluding consistency checking) was approximately 33 milliseconds; this demonstrates the effectiveness in the reduction of in the number of candidate variable bindings.

With regard to the individual techniques, the evaluation demonstrated the following: given an update of size 1, the average time to apply the completion rules to the updated summary root graph and localize the affected individuals took ap-

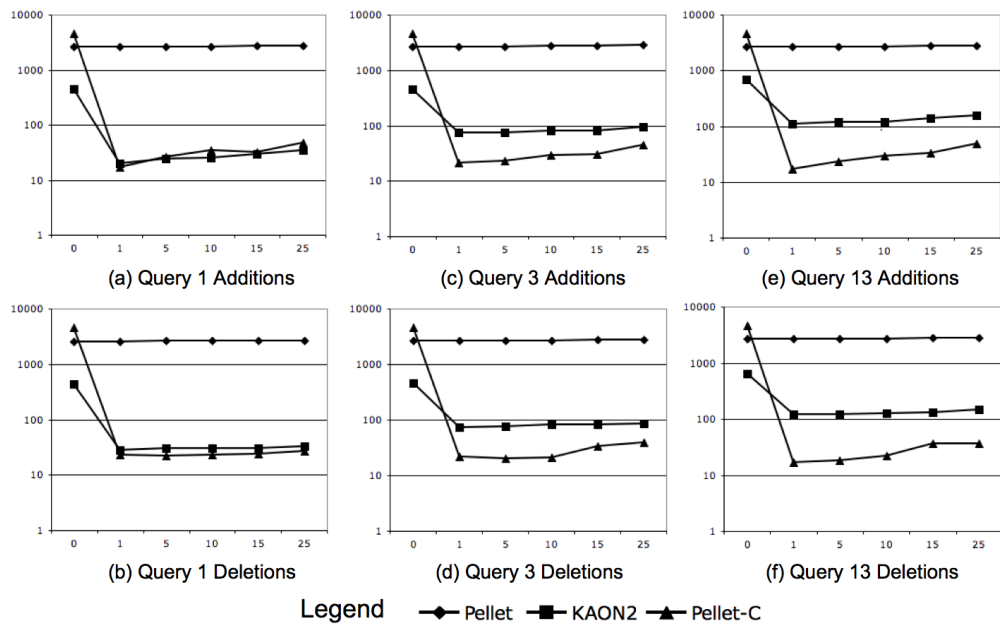


Figure 4: Continuous query answering for LUBM queries. Times (log.) in milliseconds along Y-axis. Update size along X-axis.

proximately 0.23 milliseconds. This clearly confirmed our hypothesis that the expansion rules would be applied to only a very small portion of the summary root graph. Further, this demonstrates that very little overhead is introduced when finding the individuals affected by the update. Even more promising was that the number of affected individuals was proportional to the number of individuals referenced in the update; in particular, for updates of size 1, on average, only 11.144 individuals are affected, amounting to only .068-percent of the entire KB (for increased update size, the number of affected individuals scaled proportionally to the number of individuals referenced in the update). This demonstrates a dramatic reduction in the number of individuals that needed to be considered after each update and shows that the overestimation approach may be usable in practice. Additionally, it can be seen that the optimized version of Pellet outperforms, or performs as well as, KAON2 in all cases. Even more promising is that in query 13, Pellet outperforms KAON2 by almost an order of magnitude.

6 Discussion and Future Work

Our preliminary results demonstrate that the matching approach presented in this paper can scale to a few hundred subscriptions under publish frequencies similar to that of the Dow Jones Newswire (i.e., 10,000 per day ~ approximately 7 per minute). While this may be an acceptable workload for a wide range of syndication applications (e.g., filtering financial news feeds within small to medium investment banks), for larger scale applications, additional research is necessary. One direction that we plan to investigate is leveraging the overlap and/or subsumption between registered subscriptions. General query subsumption in expressive description logics has previously been investigated (e.g., see [23]), and as future work we plan to investigate applying similar techniques to this application domain so that only a subset of the registered subscriptions will have to be re-evaluated after new information is published. Additionally, we plan to investigate to addressing the scalability of our approach is to investigate distributed OWL-based syndication frameworks (i.e., more than one broker), as this will provide increased scalability. One insight would be to potentially exploit recent work on ontology modularity for the purpose of distributing processing of subscriptions to some number of brokers.

In this paper, we have primarily addressed providing a more practical approach for finding information matches in OWL-based syndication systems. As mentioned earlier, there has been extensive work on finding minimal justifications in

OWL KBs [20]. Using such an approach, it is easy to extend information matches to publication matches. Further, initial results presented in [20] for finding justifications demonstrates that such an approach may be practical. In future work, we will explore the usage of these techniques for extending our current work.

While the evaluation presented in Section 5 demonstrates that the investigated reasoning techniques provide a more practical approach for OWL-based syndication, it is clear that other aspects of the framework need be evaluated. Also additional ontologies need to be used to further assess the effectiveness of the techniques. It is our aim to address both of these issues in future work.

We also feel that there is substantial room extending the current reasoning approaches. This includes developing additional optimizations for reasoning through changing KBs, as well as extending the current techniques to a larger portion of OWL; in particular, we feel it is certainly possible to lift the restriction that the KB be unfoldable, and we will address this in future work. Additionally, while our initial results demonstrate that the overhead of the advanced form of query impact is acceptable, we plan to further investigate the tradeoffs between the two variants presented here. We also point out here one interesting direction to explore. In particular, we plan to investigate the use of static structural analysis techniques on the broker's KB, which may allow the determination of the parts of the KB that labels can percolate. In particular, we plan to investigate recent work on structural analysis on *SHIQ* KBs for this purpose [9]. This may provide another approach for locating affected individuals.

Lastly, in real world domains, it is often the case that conflicting information is disseminated. Depending on the ontologies used within such a syndication framework, this could lead to inconsistencies. In this work we have assumed that updates which cause inconsistencies are simply discarded. Currently, we are working on developing revision techniques for OWL-DL KBs and hope to apply such efforts to resolving inconsistencies encountered in syndication systems [13].

7 Related Work

There has been substantial research on syndication systems, with a transition to more expressive approaches for representing subscription requests and published information. These have included keyword-based approaches (e.g., [26]), attribute-value pairs (e.g., [1]), XML (e.g., [2, 6]) and recently RDF-based approaches (e.g., [5, 33]). The approach presented here provides increased expressivity for representing published information (i.e., complex logic descriptions of

published content can be defined that are not representable using previous techniques).

[32, 24] proposes a DL-based approach for syndication in which DL concepts (possible complex concepts) are used for both subscription requests as well as published documents/data. [12] presents a DL-based syndication approach in which the subscriber registers queries (restricted to single, named concepts) that model their interests and published data is modeled as ABox assertions. [12] also presents two optimizations. First, the authors propose inducing a partial ordering upon all registered queries by their subsumption relations; more general queries are answered first, thereby reducing the number of individuals that must be considered for more specific queries. [12] also proposes disregarding previous individuals that satisfied registered queries when data is published. Our approach differs in a variety of ways, including that we fully formalize the approach, support complex conjunctive queries, allow subscription and published document expiration times, etc. Additionally, we address the performance bottlenecks of DL-based syndication further.

There has been substantial work in continuous query answering in relational databases and datalog (e.g., [29, 7]). While related, the work presented here addresses a more expressive formalism.

8 Conclusion

In this paper, we have formalized a OWL-based syndication framework in which DL reasoning is the primary 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 formally defined continuous queries (i.e., subscriptions) for DL KBs and presented a novel algorithm for continuous query answering. Lastly, an evaluation of the query answering approach for syndication purposes has been presented, demonstrating dramatic performance improvements.

We would like to thank Jennifer Golbeck, Yarden Katz, Vladimir Kolovski, 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, and T. D. Chandra. Matching events in a content-based subscription system. In *Symposium on Principles of Distributed Computing*, 1999.
- [2] M. Altinel and M. J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *The VLDB Journal*, 2000.
- [3] F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180, 1995.
- [4] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.
- [5] P. A. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Publish/subscribe for rdf-based p2p networks. In *In Proceedings of 1st European Semantic Web Symposium.*, 2004.
- [6] Y. Diao, S. Rizvi, and M. Franklin. Towards an internet-scale xml dissemination service. In *Proceedings of VLDB2004*, 2004.
- [7] G. Dong and R. W. Topor. Incremental evaluation of datalog queries. In *Proc. of the 4th Int. Conference on Database Theory*, 1992.
- [8] G. Flouris, D. Plexousakis, and G. Antoniou. On applying the agm theory to dls and owl. In *4th International Semantic Web Conference*, 2005.
- [9] A. Fokoue, A. Kershenbaum, C. P. Li Ma, E. Schonberg, , and K. Srinivas. Using abstract evaluation in abox reasoning. In *Int. Workshop on Scalable Semantic Web Knowledge Base Systems*, 2006.
- [10] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic *SHIQ*. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, 2007.
- [11] Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *Journal of Web Semantics*, 3(2):158–182, 2005.

- [12] V. Haarslev and R. Möller. Incremental query answering for implementing document retrieval services. In *Proc. of the Int. Workshop on Description Logics*, 2003.
- [13] C. Halaschek-Wiener, Y. Katz, and B. Parsia. Belief base revision for expressive description logics. In *In Proc. of Workshop on OWL Experiences and Directions*, 2006.
- [14] C. Halaschek-Wiener, B. Parsia, and E. Sirin. Description logic reasoning with syntactic updates. In *Proc. of the Int. Conf. on Ontologies, Databases, and Applications of Semantics*, 2006.
- [15] I. Horrocks and U. Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence*, 2005.
- [16] I. Horrocks, U. Sattler, and S. Tobies. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999.
- [17] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of the 6th Int. Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in *Lecture Notes in Artificial Intelligence*, pages 161–180. Springer-Verlag, 1999.
- [18] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proc. of the 17th National Conf. on Artificial Intelligence*, pages 399–404, 2000.
- [19] I. Horrocks and S. Tessaris. Querying the semantic web: a formal approach. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2002)*, pages 177–191, 2002.
- [20] A. Kalyanpur. Debugging and repair of owl ontologies. In *Ph.D. Dissertation, University of Maryland, College Park*.
- [21] L. Lakshmanan and S. Parthasarathy. On efficient matching of streaming xml documents and queries. In *In Extending Database Technology*, 2002.
- [22] O. Lassila and R. Swick. Resource description framework (rdf) model and syntax specification. w3c recommendation, www consortium (cambridge, ma). Feb. 1999.

- [23] A. Y. Levy and M.-C. Rousset. CARIN: A representation language combining horn rules and description logics. In *European Conf. on Artificial Intelligence*, 1996.
- [24] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. 2003.
- [25] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic aboxes. In *International Conference of Principles of Knowledge Representation and Reasoning*, 2006.
- [26] B. Oki, M. Pfluegl, and D. Skeen. The information bus: An architecture for extensible distributed systems. In *In Proc. 14th SOSP*, 1993.
- [27] M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence*, 2006.
- [28] E. Sirin. *COMBINING DESCRIPTION LOGIC REASONING WITH AI PLANNING FOR COMPOSITION OF WEB SERVICES*. PhD thesis, University of Maryland, College Park, 2006.
- [29] D. B. Terry, D. Goldberg, D. Nichols, and B. M. Oki. Continuous queries over append-only databases. In *Proc. of the Intl. Conf. on Management of Data*, 1992.
- [30] S. Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001.
- [31] D. Tsarkov and I. Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 41–50, 2004.
- [32] M. Uschold, P. Clark, F. Dickey, C. Fung, S. Smith, S. U. M. Wilke, S. Bechhofer, and I. Horrocks. A semantic infosphere. In *Proc. of the Int. Semantic Web Conference*, 2003.
- [33] J. Wang, B. Jin, and J. Li. An ontology-based publish/subscribe system. In *Middleware*, 2004.
- [34] M. Winslett. Updating logical databases. In *Updating Logical Databases*. Cambridge University Press, 1990.

9 Appendix

9.1 Theorem 1: Conditions for *SHI* Concept Instantiation

Proof. First note the following properties:

1. $Comp(K + \{\neg C(a)\}) \neq \emptyset$, as $K \not\models C(a)$
2. $Comp(K + \beta) \neq \emptyset$, as it is assumed $K + \beta$ is consistent.
3. $Comp(K + \{\neg C(a)\} + \beta) = \emptyset$, as $K + \beta \models C(a)$
4. The additional expansion rule firings caused by adding β and $\neg C(a)$ to each $G \in Comp(K)$ must cause a clash in all possible completion graphs; this follows from Theorems 1 & 2 of [14] and property 3

Consider addition updates. This case will be shown by contradiction. Assume that 1) $K \not\models C(a)$, $K + \beta \not\models \perp$, $K + \beta \models C(a)$ and 2) there does not exist $G \in Comp(K)$ 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 $\{G_1, G_2\} \subseteq Comp(K)$ s.t. the second condition of the theorem holds. Assumption 2 implies that every clash observed when updating each $G \in Comp(K)$ with $\neg C(a)$ and β is not dependent on both $\neg C(a)$ and β (note that property 4 implies a clash must occur). This implies that if the clash is dependent on β , then a clash would be observed if only β were added to G ; the same can be said for $\neg C(a)$. This in conjunction with properties 1–2 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, resulting in a contradiction. Observe that it clearly cannot be the case that every clash observed is 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, the assumption that $K + \beta \models C(a)$ 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 $\neg C(a)$. Assume this is not the case; that is, for each set of observed clashes $\{c_1, \dots, c_n\}$ that are dependent on the same non-deterministic choice, it is the case that there does not exist some c_i , $1 \leq i \leq n$, that is dependent on β and c_j , $i \neq j$, that is dependent on $\neg C(a)$. If this is the case there is a contradiction as this would imply that either $K + \beta \models \perp$ or $K \models C(a)$. Next, note that the only cause of non-determinism in the *SHI* tableau algorithm

are disjunctions; thus the observed clashes c_i and c_j must be dependent on the same disjunction. This implies that there exists $\{G_1, G_2\} \subseteq \text{Comp}(\mathbb{K})$ with the same node x with $D_1 \sqcup D_2 \in \mathcal{L}(x)$ that c_i and c_j are dependent on, and c_i and c_j are dependent on β and $\neg C(a)$ respectively. Thus, we have arrived at a contradiction.

Next consider the case for deletions; due to the assumption that $\mathbb{K} \models C(a)$ and $\mathbb{K} - \alpha \not\models C(a)$, it is the case that $(\mathbb{K} - \alpha) + \alpha \models C(a)$. Therefore, the case for deletions is a consequence of the case for additions. \square

9.2 Lemma 3: Root Paths in Completion Graphs

Proof (Sketch). This can be proven by inductively considering the application of expansion rules [17] to some concept name in the label of a node caused by the addition of C to $\mathcal{L}(a)$ and showing that the one of the two conditions in the lemma must be satisfied. Note that as G is assumed to be complete prior to adding C to $\mathcal{L}(a)$, the initial expansion rule application must be to $C \in \mathcal{L}(y)$. We simply consider the base case, as the inductive step follows in a similar manner. We address the expansion rules case-by-case:

- C is an atomic concept: this case is trivial, as none the tableau expansion rules are applicable to $C \in \mathcal{L}(a)$.
- C is of the form $A_1 \sqcap A_2$: the \sqcap -rule will add all conjuncts to the label of a . First, observe that the tableau expansion rules will not reach a new node due to application of the \sqcap -rule. Further, by definition, the modal depth of A_1 and A_2 is taken into account when determining the root path. Therefore, this case holds.
- C is of the form $A_1 \sqcup A_2$: the \sqcup -rule will add one of the disjuncts to the label of a . As in the previous case, the tableau expansion rules will not reach a new node due to application of the \sqcup -rule, implying this case holds.
- C is of the form $\forall S.A$ and there does not exist some R s.t. $\text{Trans}(R)$ and $R \sqsubseteq S$: the \forall -rule will add the concept name A to the label of all of a 's S -neighbors that do not include A in their label; assume the existence of such a S -neighbor z . By definition, the modal depth of C must be ≥ 1 , implying there is a root path to z ; note that this captures the case where z is a root node corresponding to a named individual or a non-root node.
- C is of the form $\forall S.A$ and there exists some R s.t. $\text{Trans}(R)$ and $R \sqsubseteq S$: as in the previous case, the \forall -rule will add A to all of a 's S -neighbors that do not include A in their label. This propagation follows in a similar manner as

the previous case. Given the transitive role, the \forall_+ -rule will add the concept name $\forall R.A$ to each R -neighbor z of a for all R s.t. $\text{Trans}(R)$ and $R \sqsubseteq S$. The propagation of any $\forall R.A$ concept name due to the application of the \forall_+ -rule follows in the same manner as the previous case; subsequent propagations to the neighbors of z due to a transitive role are covered, as by definition of a root path, successive traversals of transitive roles are only counted once.

- C is of the form $\exists S.A$: the \exists -rule would add a S edge from a to a new individual z and A will be added to $\mathcal{L}(z)$. By definition, the modal depth for C must be ≥ 1 ; thus, there will exist a root path to z . Clearly, subsequent rule firings can happen due to this newly added edge and node; the only non-trivial case is where the subsequent expansion rule applications cause a concept name to propagate back up this newly added edge to $\mathcal{L}(a)$, as this can then cause the propagation of a label to some other root node. We consider the following two cases:
 - Assume that there does not exist a label of the form $\forall R.D \in \mathcal{L}(a)$ where $S \sqsubseteq R$ s.t. $\forall R.D$ is not dependent on $C \in \mathcal{L}(a)$. This implies that all rule firings for labels of z and its non-root descendants are directly introduced due to and dependent on $C \in \mathcal{L}(a)$. Thus, it must be the case that if a concept B is added to the $\mathcal{L}(a)$, then $B \in \text{clos}(C)$ ⁷. This in conjunction with the definition of a root-path implies this case follows from the previous cases as a root path of depth $\text{Depth}(C)$ is considered from a and $\text{Depth}(B) \leq \text{Depth}(C)$.
 - Assume there exists a label of the form $\forall R.D \in \mathcal{L}(a)$ s.t. $S \sqsubseteq R$ where $\forall R.D$ is not dependent on $C \in \mathcal{L}(a)$. The only non-trivial case is where a did have an R neighbor prior to adding C to $\mathcal{L}(a)$ and $\forall R.D \in \mathcal{L}(a)$ contributes to the propagation of some label back to $\mathcal{L}(a)$ (otherwise the second condition of the lemma is satisfied); therefore, assume that a has an R -neighbor y prior to the addition of $C \in \mathcal{L}(a)$. Given the previous assumptions, for the existence of z to allow the propagation of some label back to $\mathcal{L}(a)$, there must be some different edge structure rooted at z which does not exist that is rooted at y . Further, it must be the case that either $A \in \mathcal{L}(z)$ or $D \in \mathcal{L}(z)$ leads to the addition of a new edge e and the other label results in the propagation of a label across e (due to the application of the \forall -rule). The case in which $A \in \mathcal{L}(z)$ leads to the addition of e implies that there must be some w in the tree rooted

⁷One exception is if $\forall S.D \in \text{clos}(C)$ and there exists some P s.t. $\text{Trans}(P)$ and $P \sqsubseteq S$; in this case, the \forall_+ -rule could add $\forall P.D$ to $\mathcal{L}(a)$. However, this does not affect correctness.

at y that does not contain an edge corresponding to e ; further, since D causes a propagation across this edge, there must exist a $\forall P.B$ concept name in $\mathcal{L}(w)$ and w does not have the necessary P -neighbor. Thus, the second condition of the lemma must hold. The case in which $D \in \mathcal{L}(z)$ leads to addition of e implies that the concept B that propagates back up e is in $\text{clos}(C)$ (with the same caveat as mentioned in the previous case). This in conjunction with the definition of a root-path implies this case follows from the previous cases, as a root path of depth $\text{Depth}(C)$ is considered from a and $\text{Depth}(B) \leq \text{Depth}(C)$.

Lastly, the case in which $C \in \mathcal{L}(a)$ invalidates a blocking condition, allowing the propagation of labels due to the original structures in G follows in a similar manner; this is implied from a) the previous cases, b) the fact that G is complete prior to the addition of C to $\mathcal{L}(a)$, c) the completeness of dynamic blocking which implies that all labels have propagated back up the trees rooted at root nodes, and d) the fact that breaking the block will simply repeatedly replicate the isomorphic sub-tree spanning from the blocking node to the blocked node until the cycle is blocked again. \square

9.3 Lemma 4: *SHI* Overestimate for Concept Instantiation

Proof. Consider additions first; this case will be shown by contradiction. Assume that $\mathbb{K} \not\models C(a)$ and $\mathbb{K} + \alpha \models C(a)$ and that neither condition in the lemma hold; that is, there does not exist $G \in \text{Comp}(\mathbb{K} + \alpha)$ s.t.

1. for some $b \in \mathcal{V}$ s.t. $D \in \mathcal{L}(b)$, D is of the form $\forall R.C$, and b does not have a P -neighbor ($P \not\sqsubseteq R$), it is the case that there is a root path of at most length $\text{Depth}(\neg C)$ between a and b
2. for some $b \in \text{EI}(\mathbb{K}, \alpha)$ there is a root path of at most length $\text{Depth}(\neg C)$ between a and b

From Theorem 1, we are given that one of the two conditions must be satisfied if $\mathbb{K} \not\models C(a)$ and $\mathbb{K} + \alpha \models C(a)$; it suffices to show that our assumptions imply that the conditions from the theorem do not hold for any $G \in \text{Comp}(\mathbb{K})$, therefore leading to a contradiction.

Consider the first condition from Theorem 1: first note that by definition, $\text{EI}(\mathbb{K}, \alpha)$ adds α to each $G \in \text{Comp}(\mathbb{K})$ and applies the necessary tableau comple-

tion rules (saturating if α enables additional non-deterministic choices). During this process, one of the following occurs:

1. extending $G \in \text{Comp}(\mathbb{K})$ with α results in a clash
2. extending $G \in \text{Comp}(\mathbb{K})$ with α does not result in a clash

The first case is trivial, as clearly any $G \in \text{Comp}(\mathbb{K})$ that satisfies the first case has a clash that is not dependent on both $\neg C(a)$ and α . Next consider the second case; observe that by adding α to all $G \in \text{Comp}(\mathbb{K})$ and saturating the necessary tableau expansion rules, the set of completions $\mathcal{G} = \text{Comp}(\mathbb{K} + \alpha)$ is obtained. By definition of $EI(\mathbb{K}, \alpha)$, during this process, one will obtain only the completions $\mathcal{G}' \subseteq \text{Comp}(\mathbb{K})$, which when updated with both α and $\neg C(a)$ do not have a clash that is only dependent on α . Given this, observe that $\neg C(a)$ and α can only jointly contribute to a clash in $G' \in \text{Comp}(\mathbb{K})$ if adding $\neg C \in \mathcal{L}(a)$ to $G \in \text{Comp}(\mathbb{K} + \alpha)$ causes a new label to be added to a node that has an edge or label added due to α . Lemma 3 implies that for this to happen, one of the two conditions in the lemma must occur involving a . Observe that if this occurs, a contradiction with our earlier assumptions will occur. Therefore, under the assumption, it cannot be the case that there can exist a $G \in \text{Comp}(\mathbb{K})$ that can satisfy the first condition of Theorem 1.

Next consider the second condition of Theorem 1: by definition of $EI(\mathbb{K}, \alpha)$, we are given all branches that have been closed directly from α ; further, we know all the labels that are added from the remaining open branches of the disjunct (this is obtained from condition 3 of Definition 13; it can be easily shown that the condition provides an over-estimate of the nodes which could have the disjunction label which has a branch that is closed by α). From the previous discussion, we have that the assumptions of the lemma imply that for all $G \in \text{Comp}(\mathbb{K})$ which when updated with α yields $G \in \text{Comp}(\mathbb{K} + \alpha)$, it cannot be the case that α and $\neg C(a)$ contribute to the clash. Further, from condition 2 of Theorem 1, it must be the case that for some $G \in \text{Comp}(\mathbb{K} + \alpha)$, $\neg C(a)$ contributes to the clash with a label that is dependent on the disjunct which is still open. As in the discussion from the first condition of Theorem 1, for this case to occur, the addition of $\neg C$ to $\mathcal{L}(a)$ must cause a label to propagate to some node $n \in EI(\mathbb{K}, \alpha)$; Lemma 3 in conjunction with our initial assumptions imply that this cannot occur. Therefore, it cannot be the case that the second condition from Theorem 1, giving rise to a contradiction.

The case for deletions can be shown in a similar manner. Assume that $\mathbb{K} \not\models C(a)$ and $\mathbb{K} + \alpha \models C(a)$ and that neither condition in the lemma hold. From

Theorem 1, we are given that one of the two conditions were satisfied when α was added to some $G \in \text{Comp}(\mathbb{K} - \alpha)$. It suffices to show that our assumptions imply that the conditions from the theorem do not hold, therefore leading to a contradiction.

Consider the first condition from Theorem 1: observe, that by soundness and completeness of axiom tracing and the approach for incremental updating completions graphs given ABox deletions (Theorem 2 of [14]), $EI(\mathbb{K}, \alpha)$ will contain all nodes which have a label and/or edge that is dependent on α from $G \in \text{Comp}(\mathbb{K})$ (by condition 2 of Definition 13). Therefore, this case can be shown in a similar manner as first condition for the case for additions.

Next consider the second condition of Theorem 1: by soundness and completeness of axiom tracing and the approach for incremental updating completions graphs given ABox deletions (Theorem 2 of [14]), $EI(\mathbb{K}, \alpha)$ will contain all nodes which have a label changes as a result of branches being re-opened due to the removal of α from \mathbb{K} , as well as the open branches before α is removed (by condition 2 of Definition 13). Therefore, this case can be shown in a similar manner as the second condition for the case for additions. \square

9.4 Lemma 5: Root Paths in Restricted Completion Graphs

Proof (Sketch). Similar to Lemma 3, this can be proven by inductively considering the application of expansion rules [17] to some concept name in the label of a node caused by the addition of C to $\mathcal{L}(a)$ and showing that the one of the two conditions in the lemma must be satisfied; again, we only consider the base case, as the inductive step follows in a similar manner. Further, we simply consider the \exists -rule, as the other cases follow in the same manner as the proof for Lemma 3. First, note that the completeness of dynamic blocking implies that in G all labels have propagated back up the trees rooted at root nodes; further, if a block were broken then it would simply repeatedly replicate the isomorphic sub-tree spanning from the blocking node to the blocked node until the cycle is blocked again. Next, observe that the assumptions of the lemma imply that a concept will never be propagated back up an edge added due to applying the \exists -rule to $C \in \mathcal{L}(a)$. Collectively, these observations imply that if a concept name B is propagated back up the newly added edge to the label of a , then $B \in \text{clos}(C)$ ⁸. This in conjunction

⁸Again, one exception is if $\forall S.D \in \text{clos}(C)$ and there exists some P s.t. $\text{Trans}(P)$ and $P \sqsubseteq S$; in this case, the \forall_+ -rule could add $\forall P.D$ to $\mathcal{L}(a)$. However, this does not affect correctness.

with the definition of a root-path implies this case follows as a root path of depth $Depth(C)$ is considered from a and $Depth(B) \leq Depth(C)$. \square

9.5 Proposition 1: *SHI* Binding Requirement

Proof. By assumption there is at least one distinguished variable in the query s.t. the query can be rolled up into the distinguished variable. Additionally, assuming the pseudo-nominal technique proposed in [18, 19], concepts can be introduced in the query for the remaining distinguished variables (thus simulating nominals), resulting in a *SHI* concept; further, because the pseudo-nominal concepts are never referenced in TBox axioms, they do not affect the modal depth of the query concept, cause any application of tableau expansion rules, or contribute to any clashes (implying that the query only needs to be rolled-up once and different substitutions for distinguished variables, and therefore pseudo-nominal concepts, are un-necessary). Therefore, this is a direct consequence of Lemma 4. \square

9.6 Lemma 7: Root Graph Containment

Proof. This will be shown by inductively considering the construction of a completion graph and summary root graph for the same KB. As the base case, consider the initial completion graph and summary root graph corresponding to the initial ABox, prior to the application of any expansion rules. Clearly, they are structurally the same, therefore the hypothesis is satisfied. Next, we address the inductive step. Assume that concept name C has just been added to $\mathcal{L}(y)$ (possibly x itself) s.t. y has unique root node x in completion graph G , and that G and S_G satisfy the hypothesis for each root node z . Let $T = Tree(x)$ in G , $G = Graph(x)$ in S_G , and $y' \in V_{S_G}$ be a node such that $y \rightarrow_{T,G} y'$ where $contain(T, G)$ holds. By induction, it must be that $C \in \mathcal{L}_{S_G}(y')$. We show that the hypothesis holds through the application of an expansion rule to $C \in \mathcal{L}(y)$ based on the form of C .

- \sqcap -rule: C_1, C_2 will be added $\mathcal{L}(y)$ if the labels do not exist. Consider two cases depending on whether y' is blocked at the time the \sqcap -rule is applied to it:
 - y' not blocked: because y' not blocked, the expansion rule will be applied when constructing S_G , maintaining the label relationship. No new edges are added, thus the containment relationship is maintained.

- y' indirectly or directly blocked: consider the case in which y' is directly blocked. The expansion rule will still be applied; thus, the label relationship will be maintained. However, we must show that a valid mapping exists as blocked nodes are not in V_G . This follows easily as there must exist some predecessor w of y' and blocking node z s.t. $\mathcal{L}_{S_G}(y') = \mathcal{L}_{S_G}(z)$; by definition of $Graph(x)$, there will be an edge from w to z with the same labels as $\langle w, y' \rangle$. This implies that y can be mapped to z rather than y' (i.e., $y \rightarrow_{T,G} z$). Because z is not blocked, the relationship will be maintained. Next, consider y' indirectly blocked; first, observe that if the application of the expansion rule to y' would cause the propagation of a label back up $\mathcal{L}_{S_G}(x)$, then the correctness of dynamic blocking would be contradicted. Next, the indirect blocking of y' implies there exists an ancestor z s.t. for some $m \in N_T$, $m \rightarrow_{T,G} z$ and z is directly blocked by an ancestor n in S_G . This and the definition of $Graph(x)$ implies that the mapping of $m \rightarrow_{T,G} z$ must be replaced with $m \rightarrow_{T,G} n$, as z and y' are blocked. Given that the block represents a cycle, there must then be a mapping $y \rightarrow_{T,G} w$ s.t. w is not blocked and w is in a path between n and z . Since w not blocked, the relationship is maintained.
- \sqcup -rule: C_1 or C_2 will be added to $\mathcal{L}(y)$; clearly multiple new completion graphs will be created. It suffices to consider one such completion graph, G'' , as the remainder follow in a similar manner. Observe that when constructing S_G , both labels of a disjunction are added to the node label. Therefore, this case follows in a similar manner as the previous case for the \sqcap -rule.
- \exists -rule: consider two cases depending on whether y' is blocked at the time the \exists -rule is applied to it:
 - y' not blocked: due to the label relationship, this rule may not be applicable to y' , as a S -neighbor w labeled with C already exists in S_G ; this case is trivial, as simply $z \rightarrow_{T,G} w$ if w is not blocked, and $z \rightarrow_{T,G} m$ if w blocked by node m . Additionally, consider the case where such a neighbor does not exist; clearly the rule will be applied to y' and again the relationship will hold.
 - y' indirectly or directly blocked: this can be shown in a similar manner as the \sqcap -rule; thus, there exists a valid mapping to z through the cycle introduced by the blocking condition and z will not be blocked. Thus, the necessary S neighbor will exist and the containment relationship will hold.

- \forall -rule: all S -neighbors z of y in G s.t. $C \notin \mathcal{L}(z)$ will have C added to $\mathcal{L}(z)$. Consider two cases:
 - z a root node: by definition of $Tree(x)$ and the tree model property of \mathcal{SHI} , it must be the case that $z = x$, as y was added to the application of the \exists -rule to a label of x . By induction, there is mapping s.t. x in S_G is a S -neighbor of y' . Observe that y' cannot be indirectly blocked, as its neighbor is a root node, implying the expansion rule will be applied. Therefore, the label relationship is maintained. Note that if y' blocked, the mapping of y into a non-blocked node follows in a similar manner as the previous cases.
 - z a non-root node: consider two cases depending on whether y' is blocked at the time the \forall -rule is applied to it: 1) y' not blocked: by induction the S -neighbor exists in S_G , and since the node is not blocked the rule will be applied. Therefore, the relationship is maintained. 2) y' indirectly or directly blocked: this follows in a similar manner as the previous case and the \sqcap -rule.
- \forall_+ -rule: This condition follows in a similar manner as the \forall -rule.

Lastly, the general case is addressed in which $y \in \mathcal{V}$ is previously blocked, yet due to the addition of a concept name C to an ancestor z of y or simply y itself, the block is invalidated resulting in the application of expansion rules to y . By induction, prior to the blocking of y , there is some y' s.t. $y \rightarrow_{T,G} y'$. We show that a valid mapping will still exist. Consider two cases: 1) y' is not blocked: by induction $\mathcal{L}(y) \subseteq \mathcal{L}_{S_G}(y')$ and the mapping still holds. Since y' not blocked, this case follows from the cases shown above. 2) y' directly or indirectly blocked: again by induction $\mathcal{L}(y) \subseteq \mathcal{L}_{S_G}(y')$ prior to applying an expansion rule to y . We must show that there exists a mapping $y \rightarrow_{T,G} z$ s.t. z not blocked; again, this can be shown similar manner to the cases shown above. \square

9.7 Lemma 8: Root Graph Update

Proof. First consider termination; this can be shown as a simple extension to the termination proof of the \mathcal{SHI} tableau algorithm [16]. Note that because the KB is expressed in \mathcal{SHI} dynamic blocking is assumed. Let $m = \#(\text{clos}(K) \cup \text{clos}(\beta))$; then termination is a consequence of the following properties:

1. The marking function θ ensures the expansion rules will be applied to a node label at most once.

2. The expansion rules never remove nodes from the summary root graph or concept from node labels.
3. Successors are only generated for existential value restrictions. For any node, during each update each of these restrictions can only trigger the generation of at most one additional successor; this is a direct consequence of the marking function. Assume there have been n previous additions. Since $\text{clos}(\mathbf{K}) \cup \text{clos}(\beta)$ cannot contain more than m existential value restrictions, the out-degree of the tree is bound by $(n + 1)m$.
4. Node are labeled with a nonempty subsets of $\text{clos}(\mathbf{K}) \cup \text{clos}(\beta)$. If a path p is of least length 2^m , then there are 2 nodes x, y of p with $\mathcal{L}(x) = \mathcal{L}(y)$, and therefore blocking occurs. Since a path on which nodes are blocked cannot become longer, paths are of length at most 2^m .

Next consider the containment relationship after the update. It is a consequence of Theorems 1 & 2 of [14], that in order to construct all complete and clash-free completion graphs for $\mathbf{K} + \beta$, one can add the structure from β to each $G \in \text{Comp}(\mathbf{K})$ and apply the necessary expansion rules. Therefore, it suffices to show that when updating each $G \in \text{Comp}(\mathbf{K})$ with β the lemma holds after updating S_G ; this can be shown as a straightforward extension of the proof for Lemma 7. Consider each update type for some $\alpha \in \beta$:

1. $\alpha = C(a)$, where a is an existing individual. By Lemma 7, $\mathcal{L}_G(x_a) \subseteq \mathcal{L}_{S_G}(x_a)$; therefore, when adding C to $\mathcal{L}(x_a)$ and $\mathcal{L}_{S_G}(x_a)$ the label relationship is clearly maintained. Consider the different expansion rules which could be applicable depending on the form of C :
 - \sqcap -rule: the expansion rule will add C_1, C_2 to $\mathcal{L}(x_a)$ if that label does not exist. By definition, C_1, C_2 will be re-added to $\mathcal{L}_{S_G}(x_a)$, and the expansion rules will be applied to both labels. Clearly, the root label relationship is maintained. Therefore, this case can inductively be shown in the same manner as in Lemma 7.
 - \sqcup -rule: the expansion rule will select C_1 or C_2 to add to $\mathcal{L}(x_a)$; clearly, multiple new completion graphs will be created, as it is assumed that all complete and clash-free completion graphs are constructed. It suffices to consider one such completion graph, G'' , as the remainder follow in a similar manner. Note that $C \in \{C_1, C_2\}$ will be added to $\mathcal{L}_{G''}(x_a)$. When updating S_G , both labels are be re-added even if they exists. Therefore, this case follows in a similar manner as the \sqcap -rule.

- \exists -rule: the expansion rule will add a new edge and node and set the appropriate labels, if x does not have an S -neighbor labeled with C . Note that root nodes cannot be blocked. Observe that due to the marking condition, a new S -successor labeled with C will be added to S_G and the expansion rule will be applied to this node; additionally, the expansion rules will be applied to the existing S -neighbors. Therefore, this case can be shown as in Lemma 7.
- \forall -rule: the concept name C will be added to the label of all of y 's S -neighbors z . Consider the case that z is a root node. By definition, the tableau expansion rules do not add edges between root nodes; therefore, there must be the corresponding edge in S_G . This, in conjunction with the marking scheme implies that the label will be re-added to the corresponding S -neighbor, and the modified expansion rules will be applied to all labels of this node. Thus, this case can be shown in a similar manner as Lemma 7. Consider the case that C is added to a non-root node $y \in \mathcal{V}$. Lemma 7 implies there is some y' in S_G where $y \rightarrow_{T,G} y'$ such that the containment is satisfied. Due to the marking scheme, the C concept name will be propagated to $\mathcal{L}(y')$. Therefore, this can be shown in a similar manner as Lemma 7.
- \forall_+ -rule: This condition follows in a similar manner as the \forall -rule.

Lastly, consider the general case in which y is previously blocked, yet due to the addition of a concept name C to an ancestor z of y , the blocking condition is invalidated resulting in the application of expansion rules to y . Given the previous cases for the various expansion rules, this case can be shown in a similar to the same case in the proof for Lemma 7.

2. $\alpha = C(a)$, where a is a new individual. This follows from Lemma 7, as the only rule applications will be for C and the universal concept C_T .
3. $\alpha = R(a, b)$. There are 4 cases to consider:
 - a and b are new individuals. This follows from Lemma 7, as the only rule applications will be those applied to a, b due to the universal concept C_T and generated existential individuals with unique root node a or b .
 - a and b are existing individuals. The only rules applicable are either the \forall -rule or \forall_+ -rule for some label in either $\mathcal{L}(a)$ or $\mathcal{L}(b)$; this is a direct consequence as G and S_G are complete prior to the update. By Lemma 7,

$\mathcal{L}_G(a) \subseteq \mathcal{L}_{S_G}(a)$ and $\mathcal{L}_G(b) \subseteq \mathcal{L}_{S_G}(b)$. Therefore, this case can be shown in the same manner as shown for these expansion rules in condition 1.

– a is a existing individual and b is an new individual. This can be shown in a similar manner as previous case.

– a is a new individual and b is an existing individual. This can be shown in a similar manner as previous case.

4. $\alpha = (a = b)$. By definition of the approach, under ABox equality updates, the completion graph nodes will be merged; then, expansion rules are applied. Note that this occurs when updating S_G . By Lemma 7, $\mathcal{L}_G(a) \subseteq \mathcal{L}_{S_G}(a)$ and $\mathcal{L}_G(b) \subseteq \mathcal{L}_{S_G}(b)$. Thus, the relationship holds for the merged node; trivially the containment relationship holds as well. Then, expansion rules are applied to all node labels of a and b . Thus, this can be shown in a similar manner as the previous cases.

5. $\alpha = (a \neq b)$. Observe that by definition of the approach, the inequality relation will be updated. The same will occur for the summary root graph. Therefore, this case trivially holds.

□

9.8 Lemma 9: Propagation in Summary Root Graph

Proof. Assume that x has concept name added to $\mathcal{L}(x)$ when updating some $G \in \text{Comp}(\mathbb{K})$ with β , yet x is not reached when applying the modified expansion rules while updating S_G . Let G' be the completion graph resulting from the update of G with β such that a concept name is added to $\mathcal{L}(x)$ and S'_G be the result of updating S_G with β . We will show that there is a contradiction by considering the different update types for some $\alpha \in \beta$ and showing that the propagation must be observed in S_G :

1. $\alpha = C(a)$, where a is an existing individual: Lemma 8 implies that after the update $\mathcal{L}_{G'}(x_a) \subseteq \mathcal{L}_{S'_G}(x_a)$. By the tree model property of *SHI*, it is the case that labels can only be propagated to some other root node by the application of the \forall -rule for some other root node. Further, by definition the tableau expansion rules do not add edges or edge labels between root nodes; so the corresponding edge must also exist in S_G . This, in conjunction with the marking scheme, implies that if the tableau expansion rules are applied

to a root node neighbor of x_a when updating G , the same expansion rule application must occur when updating S_G . The same can inductively be said for subsequently reached root nodes.

2. $\alpha = C(a)$, where a is a new individual. This follows trivially as the only rule applications will be for C and the universal concept C_T ; this in conjunction with the tree model property for \mathcal{SHI} implies the propagation holds.
3. $\alpha = R(a, b)$. There are 4 cases to consider:
 - a and b are new individuals. This follows trivially as the only rule applications will be those applied to x_a, x_b due to the universal concept C_T and generated existential individuals rooted at x_a or x_b .
 - a and b are existing individuals. Observe that the only rules applicable are either the \forall -rule or \forall_+ -rule for some label in either $\mathcal{L}(y_a)$ or $\mathcal{L}(y_b)$; this is a direct consequence as G and S_G are complete prior to the update. By Lemma 8, $\mathcal{L}_{G'}(y_a) \subseteq \mathcal{L}_{S'_G}(y_a)$ and $\mathcal{L}_{G'}(y_b) \subseteq \mathcal{L}_{S'_G}(y_b)$. Therefore, this case can be shown in the same manner as case 1.
 - a is an existing individual and b is a new individual. This can be shown in a similar manner as previous case.
 - a is a new individual and b is an existing individual. This can be shown in a similar manner as previous case.
4. $\alpha = (a = b)$. Observe that by definition of the approach, under ABox equality updates, the completion graph nodes will be merged; then expansion rules are applied. Note that this occurs when updating S_G . By Lemma 8, $\mathcal{L}_{G'}(y_a) \subseteq \mathcal{L}_{S'_G}(y_a)$ and $\mathcal{L}_{G'}(y_b) \subseteq \mathcal{L}_{S'_G}(y_b)$. Therefore, this can be shown in a similar manner as the previous cases.
5. $\alpha = (a \neq b)$. Observe that by definition of the approach, the inequality relation will be updated. The same will occur for the summary root graph. Therefore, this can be shown in a similar manner as the previous cases.

□

9.9 Lemma 10: Clashes in Summary Root Graph

Proof. Assume that there is a clash $c = (y, \neg C, C)$ caused when updating some $G \in \text{Comp}(\mathbb{K})$ with β due to the addition of $\neg C$ to $\mathcal{L}(y)$, yet either the expansion rules do not reach z or $\{\neg C, C\} \not\subseteq \mathcal{L}(z)$ from all z s.t. y can be mapped into z satisfying the containment relationship. A contradiction will be shown by considering

the different update types for some $\alpha \in \beta$ and showing that the expansion rules will reach some z and the clash will be observed.

1. $\alpha = C(a)$, where a is an existing individual. It suffices to inductively consider the different expansion rules which will be applicable depending on the form of C and show that the expansion rules will reach all z and the clash will be observed. As the base case, consider the initial addition of the concept name C to $\mathcal{L}(x_a)$. Lemma 8 implies that $\mathcal{L}(x_a) \subseteq \mathcal{L}_{S_G}(x_a)$. Therefore, if a clash is observed when adding C to $\mathcal{L}(x_a)$, it will be observed when added C to $\mathcal{L}_{S_G}(x_a)$. Because nodes are not merged, there is a unique mapping for root nodes; thus this case holds. Next consider the inductive step; assume that the hypothesis holds for all label additions thus far. Further, assume that for some node y in G with unique root node x (possibly y itself), an expansion rule is now applicable to concept name C in $\mathcal{L}(y)$. Let $T' = Tree(x)$ and $G' = Graph(x)$ at this point, and z be a node in S_G where $y \rightarrow_{T', G'} z$ s.t. the containment is satisfied. We now consider the different expansion rules applicable based on the form of C :

- \sqcap -rule: the expansion rule will add C_1, C_2 to $\mathcal{L}(y)$ if that label does not exist. Consider two cases:
 - z not blocked: By definition, C_1, C_2 will be (re)added to $\mathcal{L}_{S_G}(z)$; therefore, if the clash is observed in $\mathcal{L}(y)$, clearly it will be observed in $\mathcal{L}(z)$.
 - z directly or indirectly blocked: consider the case where z directly blocked. By definition of S_G , z is no longer a valid mapping, as it is not in V_{S_G} ; however, as shown in the proof for Lemma 8, there must exist a blocking node n s.t. $\mathcal{L}(n) = \mathcal{L}(z)$. Further, there must be some predecessor p of z . By definition, y must be mapped to n rather than z (as z blocked), resulting in the containment relationship; therefore, assume that y is mapped to n rather than z . Next observe that due to the tree model property of \mathcal{SHI} , for the rules to be applied to z , they must have been applied to n , as n is an ancestor of z . This implies the clash will be observed when the expansion rules are applied to n , as $\mathcal{L}(n) = \mathcal{L}(z)$. A similar case can be made when z indirectly blocked, as there must be some ancestor of z that is directly blocked.
- \sqcup -rule: the expansion rule will select C_1 or C_2 to add to $\mathcal{L}(y)$; clearly, multiple new completion graphs will be created. It suffices to consider

one such completion graph, G'' , as the remainder follow in a similar manner. Note that $C \in \{C_1, C_2\}$ will be added to $\mathcal{L}_{G''}(y)$. Observe that by definition of the summary root graph, both disjuncts are added to a node label if a disjunction is encountered. Therefore, this case can be shown in a similar manner as the \sqcap -rule.

- \exists -rule: if y does not have a S -neighbor labeled with C , then the expansion rule will add a new node w and edge $\langle y, w \rangle$ labeled with C and S respectively; additionally, the universal concept C_T will be added to $\mathcal{L}(w)$. Observe that no atomic clashes can occur directly as a result of this action. However, we must show that the expansion rules will subsequently be applied to all valid mapping of w into S_G . Consider the case that z not blocked. By Lemma 8, $\exists S.C \in \mathcal{L}_{S_G}(z)$; further, by definition of the marking scheme, a new S -neighbor m will be added to the summary root graph. Thus, w can trivially be mapped to m and this case holds. The cases in which z directly or indirectly blocked can be shown similar to this case and the blocked cases for the \sqcap -rule.
- \forall -rule: in G the label C will be added to all S -neighbors w of y . Consider the following two cases:
 - z not blocked: Lemma 8 implies that the corresponding S -neighbors m of z also exist in S_G s.t. $\mathcal{L}(w) \subseteq \mathcal{L}(m)$ for each $w \rightarrow_{T,G} m$. Further, the marking function implies that the label will be re-added to m . Therefore, if the clash is observed in $\mathcal{L}(w)$, clearly it will be observed in $\mathcal{L}(m)$, as $\mathcal{L}(w) \subseteq \mathcal{L}(m)$. Lastly, note that due to blocking in S_G , z itself could be mapped to w ; however, this case trivially follows as $\mathcal{L}(w) \subseteq \mathcal{L}_{S_G}(z)$ and the expansion rules are applied to $z \in \mathcal{V}_{S_G}$.
 - z directly or indirectly: this case follows in a similar manner as the blocked case for the \sqcap -rule and the previously considered case in which z is not blocked.
- \forall_+ -rule: This condition follows in a similar manner as the \forall -rule.

Lastly, consider the case where the blocking condition of a non-root node y in G is invalidated due to the addition of some concept name C to $\mathcal{L}(w)$ for some $w \in \mathcal{V}$. Observe that for there to be a clash due to the application of expansion rules to the labels in $\mathcal{L}(y)$, the blocking node z must be reached via an expansion rule application in G ; this follows from the tree-model property of SHI and completeness of dynamic blocking, which implies

further applying the expansion rules to the concept names in $\mathcal{L}(y)$ will not propagate any node label back up the tree. Further, the previously considered inductive cases for the expansion rules imply that some m in S_G , s.t. $z \rightarrow_{T,G} m$, will be reached when updating S_G . Therefore, the previous cases, Lemma 8, and the usage of the marking scheme, imply that any clashes in G observed because of newly added structures due to the breaking of the blocking condition will also be observed in S_G .

2. $\alpha = C(a)$, where a is a new individual. This follows easily from the previous cases as the only rule applications will be for C and the universal concept C_T .
3. $\alpha = R(a, b)$. There are 4 cases to consider:
 - a and b are new individuals. Again, this follows easily from the previous cases as the only rule applications will be those applied to a, b due to the universal concept C_T and generated existential individuals rooted at a or b .
 - a and b are existing individuals. Observe that the only rules applicable are either the \forall -rule or \forall_+ -rule for some label in either $\mathcal{L}(x_a)$ or $\mathcal{L}(x_b)$; this is a direct consequence as G and S_G are complete prior to the update. By Lemma 8, $\mathcal{L}_G(x_a) \subseteq \mathcal{L}_{S_G}(x_a)$ and $\mathcal{L}_G(x_b) \subseteq \mathcal{L}_{S_G}(x_b)$. Therefore, this case can be shown in the same manner as discussed above for these expansion rules.
 - a is an existing individual and b is a new individual. This can be shown in a similar manner as previous case.
 - a is a new individual and b is an existing individual. This can be shown in a similar manner as previous case.
4. $\alpha = (a = b)$. Under ABox equality updates, the completion graph nodes will be merged; then expansion rules are applied. Note that this occurs when updating S_G . By Lemma 8, $\mathcal{L}_G(x_a) \subseteq \mathcal{L}_{S_G}(x_a)$ and $\mathcal{L}_G(x_b) \subseteq \mathcal{L}_{S_G}(x_b)$. Thus, the relationship holds for the merged node. Therefore, if a clash is observed as a result of the merge in G , it will be observed in S_G . Then, expansion rules are applied to all node labels of x_a and x_b . Therefore, this case is a consequence of the previous cases.
5. $\alpha = (a \neq b)$. By definition of the approach, the inequality relation will be updated. The same will occur for the summary root graph. Therefore, this case holds as well.

□

9.10 Proposition 3: Simple Query Impact

Proof. Termination is trivial the length n must be finite and the breadth first search terminates at length n . In order to show completeness we must show that all variables binding not included in $AI_O(K, \alpha)$ must be found. Consider additions; observe that by Proposition 1 we have that for a new binding to be entailed, there must be at least one individual in the binding that is in the set of affected individuals. Further by definition of the tableau algorithm for SHI , we have that no new edges can be added between named individuals and that generated anonymous individuals can only directly be connected to one named individuals, specifically its single root. Therefore when iterating over the original affected individuals, there must be a path of less than n as by assumption all roles are simple. The approach for deletions follows directly from Proposition 1 and the monotonicity of SHI . □

9.11 Proposition 4: Mapping Query Impact

Proof. Termination is easy to show as by Proposition 2, we are given that the approach for finding $AI_O(K, \alpha)$ is terminating; further it is obvious that AI_O is a finite set of individuals, therefore termination of Query Impact can be shown.

In order to show completeness we must show that all candidate distinguished variables binding not included in $AI_O(K, \alpha)$ must be found under Query Impact. Consider additions; this follows easily as by Proposition 1 we have that for a new binding to be entailed, there must be at least one affected named individual that is bound to some variable in the query. Therefore when iterating over all mappings of the query into the completion graph with the affected individual being bound to some variable, we are guaranteed that all other candidate variable bindings must be found (which is precisely what query impact does). This is because the all bindings must be structurally mappable into all completion graphs, which follows from completeness of Theorem 3 in [27]. Again, the approach for deletions follows directly from Proposition 1 and the monotonicity of SHI . □

9.12 Proposition 5: Correctness of Algorithm 2

Proof. We first consider additions. We first note that by monotonicity of SHI , under addition only new bindings can be found; therefore we do not have to recheck previous bindings. Soundness is trivial as it follows from the soundness of the query answering algorithm provided in [19]. Termination easily follows from Proposition 2 and 4, and the termination of the query answering algorithm provided in [19]. In order to show completeness we must show that all new answer set tuples will be found. This easily follows as the algorithm simply iterates over all combinations of candidate variable bindings found by Query Impact.

Next consider deletions. Termination can be shown in a similar manner as the case for additions. Next, note that by monotonicity of SHI , under deletion only previous answer tuples can be invalidated (no new bindings can be found); therefore previous bindings need to be rechecked. In order to show soundness we must show that all tuples in the answer set after the deletion are in fact entailed. Algorithm 2 admits previous binding under two conditions; first, is when none of the bound individuals of a previous binding are in the extended set of affected individuals under simple query impact. Soundness of this follows from Proposition 3. The second case only admits previous tuples if they are entailed by [19]; therefore soundness follows from the soundness of the query answering algorithm shown in [19]. In order to show completeness we must show that only invalid previous bindings are discarded. Algorithm 2 discards previous tuples when the binding does not satisfy the query answering algorithm presented in [19]; therefore this follows from the soundness of [19]. \square