Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Diagnosing Changes in An Ontology Stream: A DL Reasoning Approach

Freddy Lécué
IBM Research, Smarter Cities Technology Centre
Damastown Industrial Estate, Dublin, Ireland
{(firstname.lastname)@ie.ibm.com}

Abstract

Recently, ontology stream reasoning has been introduced as a multidisciplinary approach, merging synergies from Artificial Intelligence, Database and World-Wide-Web to reason on semantics-augmented data streams, thus a way to answering questions on real time events. However existing approaches do not consider stream change diagnosis i.e., identification of the nature and cause of changes, where explaining the logical connection of knowledge and inferring insight on time-changing events are the main challenges. We exploit the Description Logics (DL)-based semantics of streams to tackle these challenges. Based on an analysis of stream behavior through change and inconsistency over DL axioms, we tackled change diagnosis by determining and constructing a comprehensive view on potential causes of inconsistencies. We report a large-scale evaluation of our approach in the context of live stream data from Dublin City Council.

Introduction

The *Semantic Web* (Berners-Lee, Hendler, and Lassila 2001), where the semantics of information is represented using machine-processable languages such as the *Web Ontology Language* (OWL) (OWL Working Group 2009), is considered to provide many advantages over the current formatting only version of the World-Wide-Web. OWL, for example, is underpinned by Description Logics (DL) (Baader and Nutt 2003) to define web ontologies. While this allows automatic processing of semantics-augmented data, most existing inference methods are designed for reasoning on static ontologies. However, data, information and knowledge are usually subject to change, even drastically in real world applications. From ontology versioning (Noy and Musen 2002), semantics-empowered sensors (Sheth 2010) to social semantic web (Auer, Dietzold, and Riechert 2006), all are examples of potential scenarios where knowledge is evolving over time.

Continuous collection and processing of data streams (streams for short) from sensors, actuators or social media feeds are then natural challenges that need to be addressed to answer questions on real time events. Initially tackled by the Database community (Babu and Widom 2001), these challenges have been recently exposed in the semantic web as *Stream (Ontology) Reasoning* (Valle et al. 2009) where data is semantics-tagged. Time annotated SPARQL (Rodriguez et al. 2009) or C-SPARQL (Barbieri et al. 2009) among others are potential approaches extending SPARQL, which is a syntactically-SQL-like language for querying RDF graphs, to process RDF-based data streams. Stream reasoning provides views on ontological knowledge bases, which are continuously and incrementally updated and materialized (Volz, Staab, and Motik 2005). (Ren and Pan 2011) go beyond by maintaining intermediate reasoning results and their logical relations through $\mathcal{EL}^{++}$ DL-annotated streams. These relations, reflecting dependencies in streams, together with changes are crucial to understand evolution of knowledge.

While they present efficient approaches by optimizing management of dependencies, they do not address change diagnosis over streams i.e., identification of the nature and cause of changes. So determining logical relations of reasoning results and changes over time is not considered, thus limiting explanation of their causes and impact. Even though (Kifer, Ben-David, and Gehrke 2004) and (Aggarwal 2003) provide techniques for detecting and visualizing changes, their results are subjects to interpretation and require deep analysis to identify possible logical relations between changes. Why is there a traffic jam in Dawson road at 4PM? Is it due to a music concert in Canal road at 3PM? These are general questions for which we envision change diagnosis over streams as an approach to provide insights.

We exploit the DL-based semantics of streams to tackle these challenges. Based on an analysis of stream behavior through change and inconsistency over DL axioms, we tackled change diagnosis by determining and constructing a comprehensive view on potential causes of changes. Major stream changes, modeled by axioms which are inconsistent with others at different points of time, are diagnosed by (i) identifying potential causes $\mathcal{G}$, as inconsistent axioms, given a specific window, and (ii) logically relating them by applying DL constructive reasoning *$\mathcal{G}$-guided subsumer*.

The remainder of this paper is organized as follows. Section 2 briefly reviews the logic we adopt together with stream ontology. In Section 3 we study behavior of streams through their change and inconsistency. Section 4 presents our change diagnosis approach. Section 5 reports some experiment results regarding scalability. Section 6 briefly comments on related work. Section 7 draws some conclusions.

$$Bus \sqsubseteq \exists id.DublinBusID \sqcap \exists loc.GPSLocation \quad (1)$$
$$Road \sqsubseteq \exists id.Name \sqcap \exists roadPoint.GPSSetPoint \quad (2)$$
$$Bus \sqcap \exists partOf.GPSSetPoint \sqsubseteq Road \sqcap \exists with.Bus \quad (3)$$
$$partOf \sqsubseteq roadPoint \quad (4)$$
$$JammedRoad \sqsubseteq Road \sqcap (\exists with.Bus \sqcap \exists in.HeavyTraffic) \quad (5)$$
$$FreeRoad \sqsubseteq Road \sqcap (\exists with.Bus \sqcap \exists in.LightTraffic) \quad (6)$$
$$\{C\} \sqsubseteq \exists id.\{Canal\} \sqcap \exists roadPoint.\{(53.33,-6.27),(53.33,-6.28),(53.33,-6.29)\} \quad (7)$$
$$\{D\} \sqsubseteq \exists id.\{Dawson\} \sqcap \exists roadPoint.\{(53.30,-6.25),(53.31,-6.25),(53.32,-6.25)\} \quad (8)$$
$$LightTraffic \sqcap HeavyTraffic \sqsubseteq \bot \quad (9)$$
$$ActiveEvent \sqcap NoEvent \sqsubseteq \bot \quad (10)$$
$$\{concert\} \sqsubseteq \exists loc.\{C\} \quad (11)$$

Figure 1: Sample of an $\mathcal{EL}^{++}$ TBox $\mathcal{T}$.

Background

The model we consider to represent both static background knowledge and semantics of stream data is provided by an ontology. Dynamic knowledge is then captured by reasoning on stream data descriptions in this ontology. We focus on DL as a formal knowledge representation language to define ontologies since this logic offers good reasoning support for most of its expressive families and compatibility to current W3C standards e.g., OWL 2. In the following we review (i) DL basics of $\mathcal{EL}^{++}$ and (ii) ontology stream.

$\mathcal{EL}^{++}$ Description Logics

We illustrated our work with DL $\mathcal{EL}^{++}$ where satisfiability and subsumption are decidable. The selection of this DL fragment has been guided by the expressivity which was required to model semantics of data in our domain i.e., transportation traffic and event data (Experimental Results Section). The DL $\mathcal{EL}^{++}$ (Baader, Brandt, and Lutz 2005) is the logic underpinning OWL 2 EL and the basis of many more expressive DL. Adaptations of our approach to more expressive DLs e.g., $\mathcal{SROIQ}$ (OWL 2 DL) are possible but impact decidability and complexity.

A signature Σ, defined by $(\mathcal{CN}, \mathcal{RN}, \mathcal{IN})$, consists of three disjoint sets of (i) atomic concepts $\mathcal{CN}$, (ii) atomic roles $\mathcal{RN}$, and (iii) individuals $\mathcal{IN}$. Given a signature, the top concept $\top$, the bottom concept $\bot$, an atomic concept A, an individual a, an atomic role r, $\mathcal{EL}^{++}$ concept expressions C and D can be composed with the following constructs:

$$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid \{a\}$$

We slightly abuse the notion of atomic concepts to include $\top$, $\bot$ and nominals (Horrocks and Sattler 2001) i.e., individuals appearing in concept definitions of form $\{a\}$.

The particular DL-based ontology $\mathcal{O} \doteq < \mathcal{T}, \mathcal{A} >$ is composed of a TBox $\mathcal{T}$ and ABox $\mathcal{A}$. A TBox is a set of concept and role axioms. For example, a *JammedRoad* denotes the concept of "*a road with at least a bus in a heavy traffic*" with respect to the TBox fragment illustrated in Figure 1. $\mathcal{EL}^{++}$ supports General Concept Inclusion axioms (GCIs, e.g. $C \sqsubseteq D$ with C is subsumee and D subsumer[1]) and role inclusion axioms (RIs, e.g., $r \sqsubseteq s$, $r_1 \circ \cdots \circ r_n \sqsubseteq s$). An ABox is a set of concept assertion axioms e.g., $a : C$, role assertion axioms e.g., $(a;b) : R$, and individual in/equality axioms e.g., $a \neq b$ or $a = b$. We will focus on $\mathcal{T}$ that supports concept-based inference through TBox DL reasoning.

(Baader, Brandt, and Lutz 2005) present a completion-based algorithm to classify an $\mathcal{EL}^{++}$ TBox $\mathcal{T}$ and entail subsumption for any concept in $\mathcal{CN}_{\mathcal{T}}$. Reasoning with such rules (Table 1) is PTime-Complete (Baader, Brandt, and Lutz 2008). We internalize ABox axioms into TBox axioms (through $\rightsquigarrow$) so rules in Table 1 can be applied on both axioms. Thus TBox reasoning (e.g., subsumption, satifiablility) can be also performed on internalized ABox axioms.

$$a : C \rightsquigarrow \{a\} \sqsubseteq C \qquad (a,b) : r \rightsquigarrow \{a\} \sqsubseteq \exists r.\{b\}$$
$$a \doteq b \rightsquigarrow \{a\} \equiv \{b\} \qquad a \neq b \rightsquigarrow \{a\} \sqcap \{b\} \sqsubseteq \bot$$

Besides considering internalized ABox, we assume that $\mathcal{EL}^{++}$ TBox is normalized, and all subsumption closures are pre-computed (Baader, Brandt, and Lutz 2005).

R_1	If $X \sqsubseteq A$, $A \sqsubseteq B$ then $X \sqsubseteq B$
R_2	If $X \sqsubseteq A_1, \cdots A_n$, $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$ then $X \sqsubseteq B$
R_3	If $X \sqsubseteq A$, $A \sqsubseteq \exists r.B$ then $X \sqsubseteq \exists r.B$
R_4	If $X \sqsubseteq \exists r.A$, $A \sqsubseteq A'$, $\exists r.A' \sqsubseteq B$ then $X \sqsubseteq B$
R_5	If $X \sqsubseteq \exists r.A$, $A \sqsubseteq \bot$ then $X \sqsubseteq \bot$
R_6	If $X \sqsubseteq \exists r.A$, $r \sqsubseteq s$ then $X \sqsubseteq \exists s.A$
R_7	If $X \sqsubseteq \exists r_1.A$, $A \sqsubseteq \exists r_2.B$, $r_1 \circ r_2 \sqsubseteq r_3$ then $X \sqsubseteq \exists r_3.B$

Table 1: $\mathcal{EL}^{++}$ TBox Completion Rules (no datatypes).

Ontology Stream

Contrary to (Barbieri et al. 2009) who present a framework for continuously and incrementally maintaining ontological knowledge bases, we follow (Huang and Stuckenschmidt 2005) by studying stream reasoning over a dynamic and evolutive version of ontologies i.e., ontology streams. These are more appropriate for detecting changes among versions and reasoning over their evolution. An ontology stream is considered as a sequence of ontologies (Definition 1).

Definition 1. ***(Ontology Stream)*** *An ontology stream $\mathcal{O}_m^n$ from point of time m to point of time n is a sequence of ontologies $(\mathcal{O}_m^n(m), \mathcal{O}_m^n(m+1), \cdots, \mathcal{O}_m^n(n))$ where $m,n \in \mathbb{N}$ and $m < n$.*

$\mathcal{O}_m^n(i)$ is a snapshot of an ontology stream (stream for short) $\mathcal{O}_m^n$ at point of time i, referring to a set of axioms in a DL $\mathcal{L}$. A transition from $\mathcal{O}_m^n(i)$ to $\mathcal{O}_m^n(i+1)$ is an update.

Snapshot		$\mathcal{O}_0^9(6)$		$\mathcal{O}_0^9(7)$		$\mathcal{O}_0^9(8)$	
GCI *Subsumer*	J	F		J	F	J	F
GCI *Subsumee*	-	$\{C\},\{D\}$		$\{C\}$	$\{D\}$	$\{C\}$	$\{D\}$
Completion Rule	-	R_1-R_3, R_6					
Required Axioms	-	(1-4)					
		(6-8), (12-15)		(5), (7), (19-20)	(6), (8) (17-18)	(5), (7) (24-25)	(6), (8) (22-23)

Table 2: GCIs: $\mathcal{T} \cup \mathcal{O}_0^9(i) \models Subsumee \sqsubseteq Subsumer$.

[1]For the sake of clarity we consider atomic subsumers in GCIs.

$\mathcal{O}_0^9(6)$: $Bus07 \sqsubseteq \exists id.\{dub07\} \sqcap \exists loc.\{(53.30,-6.25)\}$ (12)
: $Bus07 \sqsubseteq \exists in.LightTraffic$ (13)
: $Bus31 \sqsubseteq \exists id.\{dub31\} \sqcap \exists loc.\{(53.33,-6.27)\}$ (14)
: $Bus31 \sqsubseteq \exists in.LightTraffic$ (15)
: $\{concert\} \sqsubseteq NoEvent$ (16)

$\mathcal{O}_0^9(7)$: $Bus07 \sqsubseteq \exists id.\{dub07\} \sqcap \exists loc.\{(53.31,-6.25)\}$ (17)
: $Bus07 \sqsubseteq \exists in.LightTraffic$ (18)
: $Bus31 \sqsubseteq \exists id.\{dub31\} \sqcap \exists loc.\{(53.33,-6.28)\}$ (19)
: $Bus31 \sqsubseteq \exists in.HeavyTraffic$ (20)
: $\{concert\} \sqsubseteq NoEvent$ (21)

$\mathcal{O}_0^9(8)$: $Bus07 \sqsubseteq \exists id.\{dub07\} \sqcap \exists loc.\{(53.32,-6.25)\}$ (22)
: $Bus07 \sqsubseteq \exists in.LightTraffic$ (23)
: $Bus31 \sqsubseteq \exists id.\{dub31\} \sqcap \exists loc.\{(53.33,-6.28)\}$ (24)
: $Bus31 \sqsubseteq \exists in.HeavyTraffic$ (25)
: $\{concert\} \sqsubseteq ActiveEvent$ (26)

Figure 2: Stream Snapshots: $\mathcal{O}_0^9(6)$, $\mathcal{O}_0^9(7)$, $\mathcal{O}_0^9(8)$.

Example 1. ***(Ontology Stream)***
Figure 2 illustrates a partial ontology stream $\mathcal{O}_0^9$ through some GCIs of snapshots $\mathcal{O}_0^9(6)$, $\mathcal{O}_0^9(7)$ and $\mathcal{O}_0^9(8)$.

By applying completion rules in Table 1 on both background knowledge $\mathcal{T}$ and streams $\mathcal{O}_m^n$, we are able to infer axioms which are specific to some snapshots.

Example 2. ***(Ontology Stream and Completion Rules)***
Table 2 captures subsumption relations we can infer from axioms in $\mathcal{T}$ (Figure 1), stream $\mathcal{O}_0^9$ (Figure 2) and rules in Table 1. Inferred axioms are GCIs of the form $\mathcal{T} \cup \mathcal{O}_0^9(i) \models$ Subsumee $\sqsubseteq$ Subsumer. Table 2 depicts concepts that are subsumed by FreeRoad (F for short) and JammedRoad (i.e., J). In other words we infer that $\{C\}$ is a JammedRoad in $\mathcal{O}_0^9(7)$, $\mathcal{O}_0^9(8)$ using axioms (1-4), (5), (7)*, (19-20) and (24-25) while $\{D\}$ is a FreeRoad from $\mathcal{O}_0^9(6)$ to $\mathcal{O}_0^9(8)$. Similarly, $\{C\} \sqsubseteq F$ is entailed in $\mathcal{O}_0^9(6)$.*

Capturing the Behavior of Streams

This section identifies core properties of streams which capture the behavior of streams (required for change diagnosis) i.e., ontology stream change and inconsistency.

Ontology Stream Change

Snapshots are connected via change operations (Definition 2). Change operations from one snapshot to another one are formalized through new, obsolete and invariant GCIs.

Definition 2. ***(GCIs-based Stream Changes)***
Let $\mathcal{L}$ be a DL. $\mathcal{O}_m^n(i)$, $\mathcal{O}_m^n(j)$ be snapshots in $\mathcal{O}_m^n$. $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. GCIs-based stream changes occurring from $\mathcal{O}_m^n(i)$ to $\mathcal{O}_m^n(j)$, denoted by $\mathcal{O}_m^n(j) \backslash \mathcal{O}_m^n(i)$, are GCIs $C \sqsubseteq D$ being new (27)*, obsolete* (28) *and invariant* (29)*.*

$$\{C \sqsubseteq D \mid \mathcal{T} \cup \mathcal{O}_m^n(j) \models C \sqsubseteq D \wedge \mathcal{T} \cup \mathcal{O}_m^n(i) \not\models C \sqsubseteq D\} \quad (27)$$

$$\{C \sqsubseteq D \mid \mathcal{T} \cup \mathcal{O}_m^n(j) \not\models C \sqsubseteq D \wedge \mathcal{T} \cup \mathcal{O}_m^n(i) \models C \sqsubseteq D\} \quad (28)$$

$$\{C \sqsubseteq D \mid \mathcal{T} \cup \mathcal{O}_m^n(j) \models C \sqsubseteq D \wedge \mathcal{T} \cup \mathcal{O}_m^n(i) \models C \sqsubseteq D\} \quad (29)$$

Definition 2 evaluates GCIs-based differences among snapshots of streams. In particular it materializes knowledge changes through GCIs. This definition is a generalization of *new*, *obsolete* and *invariantChildren* introduced by (Huang and Stuckenschmidt 2005) where GCIs are considered here. Such a definition has been considered to detect more general changes. This has been motivated by complex cases where the identification of subsumption of conjuncts were required (e.g., subsumees of jammed roads, longer than 1 km and connected to more than 5 roads). (27) reflects further knowledge we obtain by moving from $\mathcal{O}_m^n(i)$ to $\mathcal{O}_m^n(j)$ while (28) denotes knowledge we sacrifice. (29) captures knowledge that remains true in both snapshots and then identifies stability of knowledge. Identifying knowledge changes is important as they provide basic properties to understand how knowledge is articulated and connected among snapshots of streams, thus a first step towards change diagnosis.

Remark 1. ***(Heterogeneity of Stream Changes)***
Although we focus on GCIs-based changes of the form $C \sqsubseteq D$ in Definition 2, it is straightforward to extend it to satisfiable conjunction $\neg(C \sqcap D \sqsubseteq \bot)$, equivalence $C \equiv D$ or disjunction $C \sqcap D \sqsubseteq \bot$ (Li and Horrocks 2003). In the latter case, the identification of a new change from $\mathcal{O}_m^n(i)$ to $\mathcal{O}_m^n(j)$ consists in finding C and D where $C \sqcap D \sqsubseteq \bot$ is true in $\mathcal{O}_m^n(j)$ but not in $\mathcal{O}_m^n(i)$. This modifies the way we detect changes, which could fit better to some applications e.g., in case subsumption is a too strong assumption.

Example 3. ***(GCIs-based Stream Changes)***
Table 3 illustrates changes (new, invariant, obsolete) occurring from $\mathcal{O}_0^9(i)$ to $\mathcal{O}_0^9(j)$ by GCIs $\{C\} \sqsubseteq J$, $\{C\} \sqsubseteq F$ and $\{D\} \sqsubseteq F$ with $6 \leq i < j \leq 8$ e.g., $\{C\}$ is a new JammedRoad in $\mathcal{O}_0^9(7)$ and $\mathcal{O}_0^9(8)$ with respect to $\mathcal{O}_0^9(6)$.

Snapshot	$\mathcal{O}_0^9(8) \backslash \mathcal{O}_0^9(7)$			$\mathcal{O}_0^9(7) \backslash \mathcal{O}_0^9(6)$			$\mathcal{O}_0^9(8) \backslash \mathcal{O}_0^9(6)$		
Changes	*new*	*inv*	*obs*	*new*	*inv*	*obs*	*new*	*inv*	*obs*
$\{C\} \sqsubseteq J$		✓		✓			✓		
$\{C\} \sqsubseteq F$						✓			✓
$\{D\} \sqsubseteq F$		✓			✓			✓	

Table 3: GCIs-based Stream Changes $\mathcal{O}_0^9(j) \backslash \mathcal{O}_0^9(i)$.

We generalize invariance axioms (29) to capture k snapshots with common GCIs (Definition 3), thus snapshots which share some knowledge. Axioms which remain true over a sequence of k snapshots are then identified.

Definition 3. ***(GCIs-based K-Invariance)***
Let $\mathcal{L}$ be a DL. C, D be two concepts in $\mathcal{L}$. $\mathcal{O}_m^n$ be a stream. $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. GCI $C \sqsubseteq D$ is k-invariant in $\mathcal{O}_m^n$ with $k \leq m-n+1$ iff $\exists i \in [m, n+1-k]$, $\forall l \in [0, k-1]$:

$$\mathcal{T} \cup \mathcal{O}_m^n(i+l) \models C \sqsubseteq D \quad (30)$$

GCIs (30) are *k-invariant GCIs* of $\mathcal{O}_m^n$ while concepts C in (30), denoted by $invariant^k\langle \mathcal{O}_m^n, \mathcal{T}, D, i\rangle$, are called *k-invariant subsumees* of D. Definition 3, as a generalization of (29) which captures 2-invariance, can be recast by $invariant(\mathcal{O}_m^n(i+l+1) \backslash \mathcal{O}_m^n(i+l))$ with same constraints on variable i and $l \in [0, k-1)$.

Example 4. ***(GCIs-based K-Invariance)***
According to Table 3, $\{D\} \sqsubseteq F$ and $\{D\}$ are respectively 3-invariant GCIs and subsumee of F. In other words $\{D\}$ is a FreeRoad from $\mathcal{O}_0^9(6)$ to $\mathcal{O}_0^9(8)$.

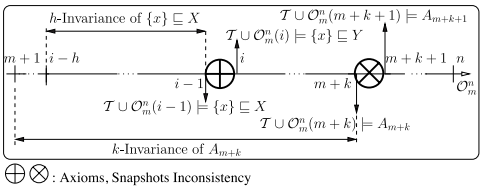

Figure 3: Illustration of Algorithm 1.

Ontology Stream Inconsistency

As mentioned in previous sections, inferring new axioms is one of the source of changes on streams (e.g., Table 2). It is common that these axioms might be inconsistent with previous snapshots as they could reflect a major disruption between two snapshots. For instance, a *free* road may become *jammed* from $\mathcal{O}_m^n(i)$ to $\mathcal{O}_m^n(j)$, but both *jammed* and *free* road cannot co-exist as a common description for any road. Such axioms are called *inconsistent axioms* (Definition 4) as they make $\mathcal{T} \cup \mathcal{O}_m^n(i) \cup \mathcal{O}_m^n(j)$ inconsistent.

Definition 4. ***(Stream-based Inconsistency of GCIs)*** *Let $\mathcal{L}$ be a DL. C, D, E be three concepts in $\mathcal{L}$. $\mathcal{O}_m^n$ be a stream. $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. GCIs $C \sqsubseteq D$ and $C \sqsubseteq E$ are inconsistent in $\mathcal{O}_m^n$ iff $\exists i, j \in [m,n], i \neq j$:*

$$\mathcal{T} \cup \mathcal{O}_m^n(i) \models C \sqsubseteq D \quad (31) \qquad \mathcal{T} \cup \mathcal{O}_m^n(j) \models C \sqsubseteq E \quad (32)$$

$$\mathcal{T} \models D \sqcap E \sqsubseteq \bot \quad (33)$$

Concepts C in Definition 4, noted $sic(\mathcal{O}_m^n, \mathcal{T}, D, E, i, j)$, are called *stream-based inconsistent concepts* while $\mathcal{O}_m^n(i)$ and $\mathcal{O}_m^n(j)$ are defined as *inconsistent snapshots* of $\mathcal{O}_m^n$.

Theorem 1. ***(Inconsistency Detection over Changes)*** *Let $\mathcal{L}$ be a DL. Let C, D, E be three concepts in $\mathcal{L}$. Let $\mathcal{T}$ and $\mathcal{O}_m^n(k)_{,m \leq k \leq n}$ be consistent terminologies and $i, j \in [m,n] : j > i$. $C \in sic(\mathcal{O}_m^n, \mathcal{T}, D, E, i, j)$ iff a) GCI $C \sqsubseteq E \in new(\mathcal{O}_m^n(j) \backslash \mathcal{O}_m^n(i))$, b) GCI $C \sqsubseteq D \in obsolete(\mathcal{O}_m^n(j) \backslash \mathcal{O}_m^n(i))$ and c)* (33).

Proof. $[\rightarrow]$ $\mathcal{T} \cup \mathcal{O}_m^n(j) \not\models C \sqsubseteq D$ due to consistent ontology $\mathcal{O}_m^n(j)$, (32), (33). Similarly $\mathcal{T} \cup \mathcal{O}_m^n(i) \not\models C \sqsubseteq E$ because of consistency of $\mathcal{O}_m^n(i)$, axioms (31) and (33). Therefore any concept in *sic* is in both $new(\mathcal{O}_m^n(j) \backslash \mathcal{O}_m^n(i), D)$ and $obsolete(\mathcal{O}_m^n(j) \backslash \mathcal{O}_m^n(i), E)$. $[\leftarrow]$ Any concept C with properties a), b) and c) is in $sic(\mathcal{O}_m^n, \mathcal{T}, D, E, i, j)$ because C is subsumed by concepts which satisfy *sic*. □

Theorem 1 ensures that inconsistencies (Definition 4) can be detected through basic stream changes (Definition 2).

Example 5. ***(Stream-based Inconsistency of GCIs)*** *Both GCIs $\{C\} \sqsubseteq F$, $\{C\} \sqsubseteq J$ are identified as inconsistent in $\mathcal{O}_0^9(6)$ and $\mathcal{O}_0^9(7)$ due to* (5), (6), (9) *in $\mathcal{T}$ and R_5 in Table 1. Thus we detect a major disruption from $\mathcal{O}_0^9(6)$ to $\mathcal{O}_0^9(7)$ i.e., free road $\{C\}$ becomes* jammed. *In other words $\{C\} \in sic(\mathcal{O}_m^n, \mathcal{T}, F, J, 6, 7)$ and $\mathcal{O}_0^9(6)$ is inconsistent with $\mathcal{O}_0^9(7)$. In the same way $\{concert\} \in sic(\mathcal{O}_m^n, \mathcal{T}, A, N, 7, 8)$ (where A, N stands for ActiveEvent NoEvent) with respect to* (10), (21), (26).

Deciding k-invariance (Definition 3) of a GCI in $\mathcal{O}_m^n(i)$ which is inconsistent with $\mathcal{O}_m^n(j)$ (Definition 4) is PTIME-hard. Indeed deciding both properties are depending on complexity of subsumption in $\mathcal{EL}^{++}$, which is PTIME-hard (Baader, Brandt, and Lutz 2005). Computing the maximal number k of snapshots wherein GCIs remains true are PSPACE-hard with respect to a polynomial k (due to a polynomial k of space required to identify its maximal).

Diagnosing Changes

We study change diagnosis through interpretation of stream inconsistency. In particular we focus on (i) determining their causes (as GCIs) over a k-window and (ii) providing a justification, as a DL concept, by means of their combination.

Determining Causes of Stream Inconsistency

It is obvious that many inconsistent axioms could be identified among snapshots due to the nature of streams. Some of them may have been caused by high-frequency sensors (Sheth, Henson, and Sahoo 2008), which provide periodic and inconsistent snapshots of the world in very short time interval e.g., loops of green and red traffic lights. Other inconsistencies may relate to GCIs that remain true over very few snapshots, hence difficult to detect their nature. We focus on inconsistencies of k-invariant GCIs i.e., k-invariant GCIs which turn out to be inconsistent in a given snapshot of a stream. This condition ensures we operate under a window of k snapshots to determine underlying relevant cause of inconsistencies, again as (inconsistent) GCIs.

Algorithm 1: Inconsistency Causes $\langle \mathcal{T}, \mathcal{O}_m^n, k, h, P \rangle$.

```
1  Input: T: a terminology. O_m^n: a stream. A_{m+k}, A_{m+k+1}:
       inconsistent axioms in O_m^n(m+k), O_m^n(m+k+1).
       A_{m+k} : k-invariant from O_m^n(m+1) to O_m^n(m+k). h: a
       minimal invariance (h ≤ k). P: a set of Pair of concepts
       (X,Y) such that T ⊨ X ⊓ Y ⊑ ⊥.
2  Result: Sol: a set of triple (G,i,h) with GCI G ∈ G being
       h-invariant and inconsistent with O_m^n(i+1) and
       i ∈ (m+1, m+k].
3  begin
4      Sol ← ∅;
5      // k-window from O_m^n(m+1) to O_m^n(m+k) is checked.
6      foreach snapshot O_m^n(i) with m+k ≥ i > m+1 do
7          // Inconsistent Axioms Detection - ⊕ in Figure 3.
8          foreach (X,Y) ∈ P do
9              // Inconsistency of O_m^n(i-1) ∪ O_m^n(i) wrt {x}.
10             if ∃{x} ∈ sic(O_m^n, T, X, Y, i-1, i) then
11                 // {x} is h-invariant subsumee of X.
12                 if {x} ∈ invariant^h(O_m^n, T, X, i-h) then
13                     Sol ← Sol ∪ ({x} ⊑ Y, i, h);
14     return Sol;
```

Algorithm 1 (illustrated by Figure 3) determines potential causes of inconsistencies of any stream update from $\mathcal{O}_m^n(m+k)$ to $\mathcal{O}_m^n(m+k+1)$ (line 6, $\otimes$ in Figure 3), with k in $(0, n-m-1]$. Causes are identified through GCIs A_{i-1} and A_i which are inconsistent in any previous snapshots of this k-window (line 10) i.e., $i \in (m+1, m+k]$. Inconsistencies are guided by P i.e., a set of inconsistent concepts (line 8, $\oplus$ in Figure 3). GCIs A_{i-1} are constrained to remain true over a minimum number of h snapshots, with $h \leq k$ (line 12). This restriction ensures A_{i-1} to be h-invariant and GCIs

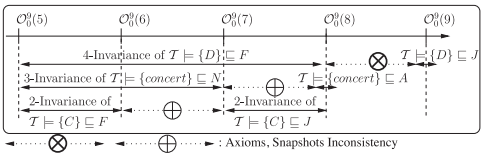

Figure 4: $\mathcal{O}_0^9(5)$ to $\mathcal{O}_0^9(9)$: Invariances and Inconsistencies.

that could be noisy sources of inconsistency are discarded. Finally, h-invariant GCIs, which are inconsistent with snapshot $i \leq m+k$, are returned as potential causes of inconsistencies of A_{m+k} and A_{m+k+1} (line 13).

Computing a solution according to Algorithm 1 in a context of polynomial input h, k and number of elements in P is a PSPACE-hard problem due to the complexity of invariance (Definition 3) and inconsistency (Definition 4) in $\mathcal{EL}^{++}$.

Example 6. ***(Inconsistency Causes)***
Let $\mathcal{O}_0^9(5)$ be similar to $\mathcal{O}_0^9(6)$, and $\mathcal{O}_0^9(8)$ be inconsistent with $\mathcal{O}_0^9(9)$ due to (9)*,* (34)*,* (35) *(illustrated in Figure 4).*

$$\mathcal{T} \cup \mathcal{O}_0^9(8) \models \{D\} \sqsubseteq F \quad (34) \qquad \mathcal{T} \cup \mathcal{O}_0^9(9) \models \{D\} \sqsubseteq J \quad (35)$$

Detecting causes of inconsistency of these GCIs in $\mathcal{O}_0^9$ consists in applying Algorithm 1 with (i) invariance of (34) *which is* 4*, (ii) inconsistent concepts for variable P e.g., concepts in* (9)*,* (10) *and (iii) an initial h e.g., $h \geq 2$. Then Algorithm 1 identifies $\{C\} \sqsubseteq J$ in $\mathcal{O}_0^9(7)$ and $\{concert\} \sqsubseteq A$ in $\mathcal{O}_0^9(8)$ as causes of inconsistency of $\mathcal{O}_0^9(8)$ and $\mathcal{O}_0^9(9)$. In other words, $\{D\}$ could turn jammed in $\mathcal{O}_0^9(9)$ because $\{C\}$ turned jammed in $\mathcal{O}_0^9(7)$ and a $\{concert\}$ started in $\mathcal{O}_0^9(8)$.*

Minimal Justification of Stream Inconsistency

We determine how causes of inconsistency, as GCIs, could be related so they provide a justification for any inconsistent GCI $C \sqsubseteq D$. This justification is determined by constructing a DL description E such that $C \sqsubseteq E$, derived from GCIs $\mathcal{G}$ (result of Algorithm 1) and axioms in $\mathcal{T}$ (Definition 5).

Definition 5. ***($\mathcal{G}$-Guided Subsumer Problem - GSP)***
Let $\mathcal{L}$ be a DL. $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. $\mathcal{G}$ be GCIs in $\mathcal{L}$. C be a concept in $\mathcal{L}$. A GCIs $\mathcal{G}$-guided Subsumer Problem, denoted as $GSP\langle \mathcal{L}, \mathcal{T}, \mathcal{G}, C\rangle$ is finding a concept description E in $\mathcal{L}$ such that (i) $\mathcal{T} \not\models C \sqsubseteq E$, (ii) $\mathcal{T} \cup \mathcal{G} \models C \sqsubseteq E$.

Definition 5 establishes a *relation* between C and E, constructed with some causes of inconsistency $\mathcal{G}$. Relation means subsumption but one can imagine other binary relations. Causes of inconsistency $G \in \mathcal{G}$ (captured by Algorithm 1) are identified as irrelevant if $\mathcal{T} \cup (\mathcal{G} \setminus \{G\}) \models C \sqsubseteq E$ for any solution E of a GCIs $\mathcal{G}$-guided subsumer problem. We use $\mathcal{P}$ as a symbol for a $GSP\langle \mathcal{L}, \mathcal{T}, \mathcal{G}, C\rangle$ and denote with $SOLGSP(\mathcal{P})$ the set of all solutions as concepts to $\mathcal{P}$.

Example 7. ***($\mathcal{G}$-Guided Subsumer Problem - GSP)***
GCIs $\mathcal{G}$, defined by $\{\{C\} \sqsubseteq J, \{concert\} \sqsubseteq A\}$, are identified as causes of inconsistency of (34)*,* (35) *in Example 6. Let $\mathcal{T}'$ be defined by* (11) *and $D \sqsubseteq \exists ad\, jRoad.\{concert\}$ where D is an adjacent road to $\{concert\}$. $SOLGSP\langle \mathcal{L}, \mathcal{T}', \mathcal{G}, D\rangle$ are (i) $\exists ad\, jRoad.A$ and (ii) $\exists ad\, jRoad.(\exists loc.J)$. The reasons for D to be congested in $\mathcal{O}_0^9(9)$ are: (i) D is adjacent to a road with an active event and (ii) D is adjacent to a jammed road. Both descriptions are derived from $\mathcal{T}' \cup \mathcal{G}$ but not from $\mathcal{T}'$ only.*

Proposition 1. ***($\mathcal{G}$-Guided Subsumer Complexity)***
Let $\mathcal{P}$ be a GSP. If concept subsumption with respect to a $\mathcal{T} \cup \mathcal{G}$ in $\mathcal{L}$ is a problem $\mathcal{C}$-hard for a complexity class $\mathcal{C}$, then deciding whether a concept is in $SOLGSP(\mathcal{P})$ is $\mathcal{C}$-hard.

Proof. Since $\mathcal{T} \not\models C \sqsubseteq E$ and $\mathcal{T} \cup \mathcal{G} \models C \sqsubseteq E$ iff $E \in SOLGSP(\mathcal{P})$, such a problem is $\mathcal{C}$-hard. □

Thus, deciding if a concept is in $SOLGSP(\mathcal{P})$ is PTIME-hard in $\mathcal{EL}^{++}$. From an another analogy with subsumption, constructing concepts in $SOLGSP(\mathcal{P})$ is PSPACE-hard.

$SOLGSP(\mathcal{P})$ is not necessarily a unique concept, especially in a context of several GCIs in $\mathcal{T} \cup \mathcal{G}$ with C as subsumee concept. Since all concepts of $SOLGSP(\mathcal{P})$ provide a generalization of concept C, we provide a minimal view of inconsistency justification, as a DL concept, through Algorithm 2. This approach aims at (i) computing their conjunction (line 7), (ii) rewriting the result following (Baader, Küsters, and Molitor 2000) (line 11) and (iii) returning its minimal form (line 13). The first step is required to expose various causes of inconsistency through a common description B. The second step rewrites B into an equivalent ($\equiv_{\mathcal{T}}$) description through $SOLRW\langle \mathcal{L}, \mathcal{T}, B, \equiv_{\mathcal{T}}\rangle$ by using some of the names defined in $\mathcal{T}$. The third step returns the shortest description based on its size. Concept descriptions are ordered by size i.e., $E \preceq E'$ iff $|E| \leq |E'|$. The size $|E|$ (Küsters 2001) of a concept description E is defined to be the number of occurrences of concepts and role names in E (where $\top$ and $\bot$ are not counted).

Algorithm 2: Minimal Justification $\langle \mathcal{L}, \mathcal{T}, \mathcal{G}, C\rangle$.

```
1  Input: L: a DL. T: a terminology. G: a set of GCIs in L. C: a
       concept in L
2  Result: Minimal justification J ∈ L of inconsistency C ⊑ D.
3  begin
4      // Initialization of a temporary concept B ∈ L.
5      B ← ⊤;
6      // Conjunction of existing SOLGSP⟨L, T, G, C⟩.
7      foreach S_i ∈ SOLGSP⟨L, T, G, C⟩ do B ← B ⊓ S_i;
8      // Initialization of justification J.
9      J ← B;
10     // Rewriting Problem.
11     foreach S_i ∈ SOLRW⟨L, T, B, ≡_T⟩ do
12         // Minimal Solution.
13         if S_i ⪯ J then J ← S_i;
14     return J;
```

Finding a minimal justification (Algorithm 2) is PSPACE-hard due to PTIME-hardness of line 7 and PSPACE-hardness of line 11 (Baader, Küsters, and Molitor 2000) in $\mathcal{EL}^{++}$.

Example 8. ***(Minimal Justification)***
According to Algorithm 2 and Example 7, a minimal justification of $\langle \mathcal{L}, \mathcal{T}', \mathcal{G}, D\rangle$ is $\exists ad\, jRoad.(A \sqcap (\exists loc.J))$.

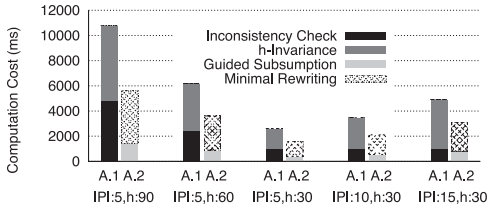

Figure 5: Computation Performance of Change Diagnosis.

Experimental Results

We report a large-scale evaluation for diagnosing changes by evaluating performance of (i) identifying GCIs which are responsible of stream inconsistencies (Algorithm 1: A.1) and (ii) constructing their minimal justification (Algorithm 2: A.2). Based on a DL extension of InfoSphere Streams (Biem et al. 2010) to process semantic streams, coupled with CEL DL reasoner (http://lat.inf.tu-dresden.de/systems/cel/), we study the impact of variables h and the number $|P|$ of inconsistent pairs of concepts e.g., axiom (10). The experiments have been conducted on server of 4 Intel(R) Xeon(R) X5650, 2.67GHz cores and 6GB RAM.

• **Context:** Stream data from the bus operator of Dublin City has been enriched with an $\mathcal{EL}^{++}$ version of the XML format SIRI (Service Interface for Real Time Information) where GPS locations and delays of 1000 buses (updated every 20 seconds) were axiomatized in 2000 GCIs. Events related to Dublin City were captured through *Eventful (http://eventful.com)* and *EventBrite (http://www.eventbrite.com)* where an average of 187 events a day have been described by 748 GCIs. Besides a core static ontology of 55 concepts with 19 role descriptions (17 concepts subsume the 38 remaining ones with a maximal depth of 3), we inject 300 $\mathcal{EL}^{++}$ GCIs to describe 100 roads and their interconnections in Dublin City.

• **Results:** Figure 5 illustrates the computation costs of the important steps for diagnosing changes over streams of $k = 1,080$ snapshots (6 hours). We considered a number of $|P| \in \{5,10,15\}$ inconsistent pairs of concepts within an exact window of $h \in 30,60,90$ snapshots (i.e., 10, 20 and 30 minutes). Despite the large number of subsumption tests required in both algorithms, our approach performs in a suitable range of computation time: from 4.2 to 16.4 seconds. Computation of Algorithm 1 represents 64% of the overall approach. It is mainly impacted by h-invariance as it requires to check subsumption over a window of up to 90 snapshots in a stream of 1,080. It is also obvious that a higher number of h and $|P|$ impacts performance of Algorithm 2. Indeed the more solutions returned by Algorithm 1 over a h-window the more inputs for Algorithm 2. The rewriting process represents 75% of Algorithm 2 and 30% of the overall approach (due to a large number of concept definition in $\mathcal{T}$). On contrary, GCIs-guided subsumption is 10% of our approach. We also note that computation time of the overall approach is much more altered by window h rather than $|P|$ e.g., by tripling $|h|$ the computation time is tripled while it is doubled when $|P|$ is tripled.

• **Validation and Open Issues:** The performance is mainly impacted by the expressivity of the DL used, e.g., the latter results would have been altered using DL $\mathcal{ALC}$ since deciding subsumption is NP-complete. However the computation performance can be improved by reducing the number of intermediate axioms in Figures 1 and 2 and (used) rules in Table 1 which are required to infer stream-based inconsistencies (Definition 4). For instance we could store only jammed and free roads axioms as DL axioms, where the latter could be "directly" derived from raw data without semantic-ization process. The size and the structure of the ontology have a limited impact compared to variables k, h and $|P|$. Besides altering computation performance, these variables have a high impact on precision. Adding too much irrelevant information i.e., noise (e.g., through too wide or narrow windows h,k) decreases precision and increases computation time. Similarly, the more data sources (modeled as GCIs) the more specific the solution, so precision and relevance of data sources are closely related.

Related Work

Continuously collected data streams are generated by dynamic processes through sensors, actuators or social media feeds. Since data changes over time, even drastically in some cases, catching the knowledge in such a dynamic environment and inferring new facts in real time are not straightforward tasks to achieve. This challenge, also known as continuous processing of information flows (i.e. data streams), has been widely investigated in the Database community (Biem et al. 2010) and more recently in the context of the *Semantic Web* (Valle et al. 2009). For instance, (Barbieri et al. 2009) presents a stream data reasoning approach, as an extension of SPARQL, to capture knowledge from multiple semantic-tagged data sources in real time. Reasoning is processed on both a rich background and evolving ontology, where the latter is materialized at query time. However, this approach is limited to relatively simple languages such as RDF and RDFS, reducing expressivity of reasoning.

(Ren and Pan 2011) address this issue by presenting a stream reasoning approach in the context of in $\mathcal{EL}^{++}$ DL. They consider streams of ontologies and focus on detecting dependencies over time. Other approaches consider changes over streams for continuously and incrementally updating and materializing knowledge. (Volz, Staab, and Motik 2005) extended the Delete and Re-derive (DRed) algorithms (Gupta, Mumick, and Subrahmanian 1993) from traditional data stream management systems for maintaining up-to-date view of knowledge. When change occurs, they overestimate the consequences of the deletion, then cashback the over-deleted consequences that can be derived by other facts, and add entailments derived from new facts. Even though some optimizations (Barbieri et al. 2010) have been proposed to better support scalable reasoning within fixed time windows, these approaches do not address change diagnosis i.e., capturing the nature of change over streams. In other context (Fanizzi, d'Amato, and Esposito 2008) address change of concept description and novelty detection using unsupervised machine learning techniques, which is based on clustering of new individuals.

(Noy and Musen 2002) detect and display changes for comparing ontology versions. In this context, changes are

likely to appear randomly in any part of the ontology, making diagnosis difficult and irrelevant. Similarly, (Huang and Stuckenschmidt 2005) address ontology versioning. They develop a temporal logic-based approach for reasoning about commonalities and differences between versions. Complying with existing DL reasoners, they are able to predict consequences of a change. However, elaborating relations between multiple changes over a large number of snapshots is out of their scope.

From the Database community perspective, changes detection has been addressed due to its impact on data processing algorithms. Indeed stable distribution of data is better handled by stream processing. Various distance metrics have been introduced for determining precisely when and how the underlying distribution changes. (Aggarwal 2003) goes further by addressing change diagnosis through understanding, visualization and identification of trend in data streams. They provide methods, based on velocity density estimation, to visualize how the pattern of the data in various spatial locations has changed over time. However trend results are then subjects to interpretation and require deeper analysis to identify possible logical relations between changes.

Conclusion and Future Work

We studied ontology stream reasoning and addressed change diagnosis i.e., identification of the nature and cause of changes over stream. While the former is important for (i) understanding how knowledge is evolving and (ii) answering questions on real time events, the latter is crucial for (iii) justifying logical connections of knowledge at various points of time and (iv) providing insight on time-changing events. Based on an analysis of stream behavior through change and inconsistency over DL axioms, we tackled change diagnosis by determining and constructing a comprehensive view on potential causes of inconsistencies.

In future work, we will further evaluate the impact of more data sources on precision and scalability. We will also adapt our approach to the Linked Open Data-based semantic web (Bizer, Heath, and Berners-Lee 2009), that will benefit from a scalability point of view.

References

Aggarwal, C. C. 2003. A framework for change diagnosis of data streams. In *SIGMOD Conference*, 575–586.

Auer, S.; Dietzold, S.; and Riechert, T. 2006. Ontowiki - a tool for social, semantic collaboration. In *ISWC*, 736–749.

Baader, F., and Nutt, W. 2003. In *The Description Logic Handbook: Theory, Implementation, and Applications*.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the el envelope. In *IJCAI*, 364–369.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the el envelope further. In *OWLED*.

Baader, F.; Küsters, R.; and Molitor, R. 2000. Rewriting concepts using terminologies. In *KR*, 297–308.

Babu, S., and Widom, J. 2001. Continuous queries over data streams. *SIGMOD Record* 30(3):109–120.

Barbieri, D. F.; Braga, D.; Ceri, S.; Valle, E. D.; and Grossniklaus, M. 2009. C-sparql: Sparql for continuous querying. In *WWW*, 1061–1062.

Barbieri, D. F.; Braga, D.; Ceri, S.; Valle, E. D.; and Grossniklaus, M. 2010. Incremental reasoning on streams and rich background knowledge. In *ESWC (1)*, 1–15.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The semantic web. *Scientific American* 284(5):34–43.

Biem, A.; Bouillet, E.; Feng, H.; Ranganathan, A.; Riabov, A.; Verscheure, O.; Koutsopoulos, H. N.; and Moran, C. 2010. Ibm infosphere streams for scalable, real-time, intelligent transportation services. In *SIGMOD*, 1093–1104.

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.* 5(3):1–22.

Fanizzi, N.; d'Amato, C.; and Esposito, F. 2008. Conceptual clustering and its application to concept drift and novelty detection. In *ESWC*, 318–332.

Gupta, A.; Mumick, I. S.; and Subrahmanian, V. S. 1993. Maintaining views incrementally. In *SIGMOD*, 157–166.

Horrocks, I., and Sattler, U. 2001. Ontology reasoning in the shoq(d) description logic. In *IJCAI*, 199–204.

Huang, Z., and Stuckenschmidt, H. 2005. Reasoning with multi-version ontologies: A temporal logic approach. In *ISWC*, 398–412.

Kifer, D.; Ben-David, S.; and Gehrke, J. 2004. Detecting change in data streams. In *VLDB*, 180–191.

Küsters, R. 2001. *Non-Standard Inferences in Description Logics*, volume 2100 of *LNCS*. Springer.

Li, L., and Horrocks, I. 2003. A software framework for matchmaking based on semantic web technology. In *WWW*, 331–339.

Noy, N. F., and Musen, M. A. 2002. Promptdiff: A fixed-point algorithm for comparing ontology versions. In *AAAI/IAAI*, 744–750.

OWL Working Group, W. 2009. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation.

Ren, Y., and Pan, J. Z. 2011. Optimising ontology stream reasoning with truth maintenance system. In *CIKM*, 831–836.

Rodriguez, A.; McGrath, R. E.; Liu, Y.; and Myers, J. D. 2009. Semantic management of streaming data. In *International Workshop on Semantic Sensor Networks at ISWC*.

Sheth, A. P.; Henson, C. A.; and Sahoo, S. S. 2008. Semantic sensor web. *IEEE Internet Computing* 12(4):78–83.

Sheth, A. P. 2010. Computing for human experience: Semantics-empowered sensors, services, and social computing on the ubiquitous web. *IEEE Internet Computing* 14(1):88–91.

Valle, E. D.; Ceri, S.; van Harmelen, F.; and Fensel, D. 2009. It's a streaming world! reasoning upon rapidly changing information. *IEEE Intelligent Systems* 24(6):83–89.

Volz, R.; Staab, S.; and Motik, B. 2005. Incrementally maintaining materializations of ontologies stored in logic databases. *J. Data Semantics* 2:1–34.