Expert Systems With Applications 90 (2017) 464–483

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

OBDAIR: Ontology-Based Distributed framework for Accessing, Integrating and Reasoning with data in disparate data sources

Georgios Santipantakis*, Konstantinos Kotis, George A. Vouros

Department of Digital Systems, University of Piraeus, Greece

ARTICLE INFO

Article history:
Received 8 December 2016
Revised 26 July 2017
Accepted 15 August 2017
Available online 24 August 2017

Keywords:
Ontology-based data access
Ontology-based data integration
Distributed ontology
Distributed reasoning
Ontology-based event recognition

ABSTRACT

The correlated exploitation of disparate and heterogeneous data sources is important to the efficacy of many analytics tasks. Currently in application domains of major interest, such as in the maritime and aviation domains, available technology provides real time surveillance data from moving entities, which together with archival static data, can be processed in an integrated way to detect complex events and support decision making. The variety of data in disparate sources, the heterogeneity of data formats, as well as the volume of data, make data retrieval, integration, and especially reasoning with these data, challenging tasks. This paper presents an ontology-based distributed framework that addresses conjunctively these challenges: Data retrieval, integration and reasoning with data from heterogeneous static or regularly updated data sources. The proposed OBDAIR framework provides the means to support building scalable data-driven domain-specific applications that support decision-making and problem-solving. This is achieved by processing large volumes of heterogeneous data close to the sources, supporting knowledge generation in a distributed/decentralized but still unified manner. OBDAIR integrates modular ontology representation frameworks and ontology-based data access frameworks: This article presents an instantiation of OBDAIR using the modular ontology representation framework $E-\mathcal{SHIQ}$, and the Ontop ontology-based access system. This OBDAIR instance has been evaluated at recognising important complex events in the maritime domain using real-world data. Experiments show the potential of OBDAIR to detect complex events in large geographic areas with computational efficiency.

1. Introduction

The correlated exploitation of data requires accessing large amounts of data in disparate sources, integrating them towards providing enriched, coherent and integrated views of entities, situations, etc. at any time, further supporting the effective, informed performance of processing tasks (e.g. analytics and reasoning tasks).

This is important in domains where tracking the trajectories of moving objects is critical, such as the maritime and aviation domains. The early recognition of events concerning vessels and airplanes is important to safety, cost, dependability, credibility and environmental-friendliness of operations. For example, the detection of events that may result to dangerous or disastrous situations in the maritime domain (Chen & Nugent, 2009; Vandecasteele & Napoli, 2012a; 2012b) is a crucial functionality for systems aiming to provide situation awareness, supporting the prevention of vessel accidents, reducing financial cost for shipping companies, and averting irrevocable damages to maritime ecosystems. However, information to support these tasks exists in disparate data sources, which are maintained and evolved autonomously.

Nevertheless, data access, retrieval and integration from disparate sources present well-known challenges in the general case: Heterogeneity of data either at the data level or at the conceptual level, the large volume of data to be retrieved, and the high frequency of dynamic data updates contribute to that.

In addition to the above, tasks such as the recognition of complex events -which is the processing task considered in this article- requires expressive formalisms for the representation of event patterns, and efficient reasoning techniques to deal with the information/data available: According to an ontology-based approach, knowledge about complex events is conceptualized by means of classes of event patterns, incorporating low-level events. Low-level events can be directly detected from: (a) raw surveillance data (e.g. low speed of vessels at a time and space instant), (b) the combination of raw surveillance data concerning distinct entities (e.g. vessels close to each other in a time interval),

* Corresponding author.
E-mail addresses: gsant@unipi.gr (G. Santipantakis), kotis.kostas@gmail.com (K. Kotis), georgev@unipi.gr (G.A. Vouros).

http://dx.doi.org/10.1016/j.eswa.2017.08.031
0957-4174/

and (c) the combination of raw surveillance data with contextual information about domain-specific classes of entities (e.g. a fishing vessel, a fishing area, a restricted area). Thus, the evaluation of patterns of complex events requires reasoning with expressive ontologies that are populated by means of information extracted from disparate, heterogeneous data sources. Having said this, it must be clarified that the aim of this paper is not to propose an ontology-based approach for the effective recognition of events, per se: This would require additional reasoning abilities about spatial and temporal features of moving objects and contextual entities, which is beyond the scope of this article.

The critical issues that the proposed framework addresses conjunctively are as follows:

- Accessing data from disparate and heterogeneous data sources, populating a modular ontology,
- Integrating data from disparate sources so as to provide coherent, consolidated and up-to-date information about moving objects,
- Reasoning with expressive ontologies that have been populated by a large amount of information.

Most of the current paradigms for accessing data sources by means of ontologies use a single ontology. The practice of using a single, monolithic ontology to represent, manage and reason with data from different data sources, presents limitations to efficiently exploit large datasets, especially when these are dynamically updated. Further criticisms to that approach are discussed in subsequent sections. On the other hand, the use of multiple ontologies presents many advantages due to the distribution of data, knowledge and tasks it may allow, but it requires the coupling of ontologies (both, at the assertional and the conceptual levels), so as to enable unified data interpretations across the distributed knowledge base.

The Ontology-based Distributed framework for Accessing, Integrating and Reasoning with data (OBDAIR) in heterogeneous and disparate data sources proposed in this article, integrates a modular ontology representation framework with an ontology-based data access framework, so as to support (a) the autonomous treatment of any data source, independently from other sources, (b) the treatment of semantically transformed data in an integrated manner, providing a framework for incorporating data integration solutions, (c) the distribution of OBDA tasks supporting the population of multiple ontologies, and finally, (d) reasoning with data in a distributed way. To balance between the efficiency of retrieving data and the efficiency of reasoning with the data, OBDAIR stresses the need to process data close to the data sources, while they are retrieved, so as to extract information which is necessary for the recognition of events: Ontologies are populated with this information, rather than with raw data. This paper demonstrates and evaluates the application of the proposed framework in the maritime domain. Evaluation aims to show the scalability of the overall approach and the potential to support near-to-real-time reasoning tasks.

The contributions of the proposed OBDAIR framework can be summarized as follows:

- Supports the use of distributed knowledge bases (modular ontologies) for retrieving, integrating and reasoning with data from disparate and heterogeneous data sources.
- Inherently enables distribution of data retrieval, integration and reasoning tasks.
- Incorporates methods for processing data on retrieval, so as to populate ontologies with information that is necessary to the recognition of complex events (in contrast to raw data), saving reasoning time.

It must be pointed out that the presented framework does not replicate data in additional stores, neither it requires to grant permission rights to alter contents of data sources. Finally, the proposed framework enables reasoning beyond the OWL2QL profile (Motik, Grau, Horrocks, Z. Wu, & Lutz, 2015) that OBDA frameworks use, towards the recognition of complex events.

The paper is an extended version of our previously published work (Santipantakis, Kotis, & Vouros, 2015a). More specifically, the extension concerns (a) the detailed description of the technologies that have been integrated in the framework and of the rationale behind these choices, (b) the detailed description of the techniques used to support the processing of data, and more importantly, (c) an extended evaluation of the proposed framework, to support our claims on scalability and near-to-real-time reasoning with the data. Additionally, the impact, implications and limitations of the proposal are discussed in depth.

The paper is structured as follows: Preliminaries on OBDA, the modular ontology representation frameworks and $E-\mathcal{SHIQ}$ are presented in Section 2. The overall architecture of the OBDAIR is presented in Section 3 and its application to the marine traffic domain is presented in Section 4. Section 5 presents and discusses experimental results on real-world data sets showing the efficiency of the proposed framework. Related work is reported in Section 6, and implications, impact and limitations of OBDAIR are thoroughly discussed in Section 7. The paper concludes with proposals for further work in Section 8.

2. Preliminaries

2.1. Single, multiple and hybrid ontology-based data access approaches

Ontologies, as formal models of representation with explicitly defined concepts and named relationships linking concept instances, provide the means to address the issue of semantic heterogeneity of data sources. It may be useful to recall the role ontologies can play:

- Enable the accurate interpretation of data from multiple sources through the explicit definition of terms and relationships, describing data about specific entities and their relations.
- Facilitate the formulation of queries about entities using a specific vocabulary (i.e. using the ontology as a global query schema for ontology-mediated query formulation).
- Verify the data-to-ontology mappings for integrating data from multiple sources. Mappings may be either user-specified or be generated automatically.
- Verify the final consistency of the data w.r.t. the ontology specifications.

Regarding the number of ontologies used, there are three types of ontology-based data access approaches: The single ontology, the multiple ontologies and the hybrid approaches.

The single ontology approach (e.g. Brüggemann, Bereta, Xiao, & Koubarakis, 2016; Haase et al., 2013; Vandecasteele & Napoli, 2012a; Zamboulis, Poulovassilis, & Roussos, 2008) uses one global ontology as a shared vocabulary for semantically describing data in all sources and for the semantic specification of the query vocabulary. One may argue that the global ontology can be considered as a combination of several specialized ontologies, i.e. usually one per data source. This however leads to a potentially large ontology, given the assumption that all the sources share the same point of view of the domain. Sometimes it is difficult to devise a global ontology in a way that data from disparate sources are integrated into a valid and consistent knowledge base. Moreover, this approach is susceptible to changes in data sources, which may affect the conceptualization of the domain, implying changes to the

global ontology, and affecting also the mappings to any of the data sources.

A multiple ontologies approach may use one ontology per data source (e.g. Figueiredo, Pitta, Salgado, & Souza, 2013; Mena, Illarramendi, Kashyap, & Sheth, 2000). Correspondences between these ontologies must be specified for integrating knowledge into a coherent and semantically valid knowledge base. Such an approach allows new data sources to be added in the system in a flexible way, while supporting modularity and robustness to changes on the data sources. An important benefit of this approach is that it may allow different point of views on data from each of the ontologies w.r.t. the semantics of the inter-ontology correspondences. Handling this kind of subjectivity is a challenging issue, albeit important, since it allows modelling the views that different stakeholders may have for specific situations.

The hybrid approaches use multiple ontologies that subscribe to a common top-level vocabulary (e.g. Goh, 1996). Similarly to the single ontology approach, the top-level ontology defines the domain terms and also available for queries. This approach however can be perceived as a special case of a multiple-ontologies approach, where only one ontology can be used for querying and exploiting the integrated knowledge.

The proposed framework follows the multiple ontologies approach for ontology-based data access, as a flexible and robust solution, inherently supporting distribution of knowledge, data and tasks. This choice is justified by the fact that at any time a new data source may need to be added, according to evolving requirements: This is particularly true in open settings with multiple data sources. In such settings data sources are being developed/maintained/evolved independently from others, while owners may hold subjective beliefs concerning "bridging" heterogeneity and coupling their data/information with the data/information provided by others. The subjectivity of these beliefs is an important pragmatic aspect, as different parties (data/information sources owners, software agents) may have a partial or abstract view of the data/information possessed by others, or they may not agree on the way they jointly shape information. These concerns corroborate that a multiple ontologies approach should be preferred towards a generic solution.

2.2. Modular ontology representation frameworks

To combine information in open and inherently distributed settings, we need special formalisms that take into account the complementarity and heterogeneity of independently evolved contexts corresponding to different data sources: We consider that for each data source, data are described by a local theory (context), which is defined by at-least one specific ontology. The combination of these ontologies, *ontology units*, or simply *units*, into a distributed knowledge base, provides a unifying theory for accessing and reasoning with the distributed data.

Generally, an ontology contains axioms for concepts, individuals and roles that state specific relations between these individuals. Let N_C, N_R and N_O be the mutually disjoint sets of concept, role and individual names of an ontology, respectively. The finite set of general concept inclusion (GCI) axioms of the form $C \sqsubseteq D$, where C, D are concepts, define the TBox of the ontology, denoted by **T**. The finite set of role inclusion axioms of the form $R \sqsubseteq S$ and transitivity axioms for roles define the role box **R** of the ontology, where $\{R, S\} \subseteq N_R$. The finite set of assertions of the form a: C, (a, b): R, $a \neq b$, where $a, b \in N_O$, R a possibly inverse role and C a concept, define the ABox of the ontology.

In the past years several modular ontology formalisms have been proposed. In this paper we focus to those proposals that (a) support expressive fragments of Description Logics (DL), i.e. at least $\mathcal{SHIQ}$, (b) provide constructors to establish inter-unit connections, (c) there is an available implementation of a distributed reasoner. $\mathcal{E}-$*Connections* (Grau, Parsia, & Sirin, 2004; Kutz, Lutz, Wolter, & Zakharyaschev, 2003), *Package-Based Description Logics* (Bao, Voutsadakis, Slutzki, & Honavar, 2009), *Distributed Description Logics (DDL)* (Serafini & Tamilin, 2004), *Integrated Distributed Description Logics (IDDL)* (Zimmermann, 2007) and $E-\mathcal{SHIQ}$ (Santipantakis & Vouros, 2014), are the frameworks considered under the scope and requirements of the proposed framework. A comprehensive and detailed reference to the background of modular representation formalisms for Description Logics is given in Zimmermann (2013).

2.2.1. The $\mathcal{E}-$Connections

The $\mathcal{E}-$*Connections* (Grau et al., 2004; Kutz et al., 2003) knowledge representation formalism, assuming disjoint domains for different units, allows ontology units to be interpreted distinctly, and enables inter-unit connections using special terms called "links". Links are similar to roles in Description Logics, relating instances in different units. Restrictions can be associated to links. Assuming disjoint domains of coupled units, $\mathcal{E}-$*Connections* confines the data sources that can be combined.

2.2.2. The Distributed Description Logics (DDL)

The Distributed Description Logics (DDL) (Serafini & Tamilin, 2004) extends standard DL with cross-units bridge rules. A variety of adjustments on DDL semantics have been proposed (Ghidini, Serafini, & Tessaris, 2007; Serafini, Borgida, & Tamilin, 2005; Serafini & Tamilin, 2005a; 2005b) so far. However, the basic bridge rules express concept-to-concept, role-to-role and individual correspondences, between ontology units. Specifically, given a non-empty set of indices I and a collection of units indexed by I, a concept-to-concept (resp. role-to-role) bridge rule $i : X \xrightarrow{\sqsubseteq} j : Y$, between concepts (resp. roles) of units i, j, specify that from the "point of view" of unit j concept (resp. role) Y is a super-concept (resp. super-role) of X. Similarly, bridge rule $i : X \xrightarrow{\sqsupseteq} j : Y$ expresses that Y is a subconcept (resp. subrole) of X, from the point of view of j. Consequently, DDL can be used only in cases where domains of distinct ontologies overlap. In addition to this, specific constraints to correspondences have been proposed to preserve decidability and to support propagation of subsumptions among units (Homola & Serafini, 2010).

2.2.3. The Package-Based Description Logics (P-DL)

The *Package-Based Description Logics (P-DL)* (Bao et al., 2009) extends standard DL with inter-unit associations in the form of semantic imports of terms. Imports are satisfied when the local interpretation of the imported terms are the same in the importing and imported ontologies. Such inter-unit associations between terms express domain relations as the bridge rules in DDL. These, under specific conditions support the propagation of implications among units. P-DL can be used only in cases where domains of distinct ontologies overlap, but in contrast to DDL, P-DL do not support subjectivity of terms' associations.

2.2.4. Integrated Distributed Description Logics (IDDL)

The Integrated Distributed Description Logics (IDDL) (Zimmermann, 2007) share similar issues to DDL and P-DL, but also it has considerable differences on the inter-unit associations and their topology. Specifically, IDDL employs correspondences between elements of different units similarly to DDL, but from the point of view of a "third" unit, which coordinates the associated units. The associations between units are bi-directional and equally affect the related units. This approach allows the reasoning framework to handle the correspondences as standard DL operators. Thus, IDDL can be used for federated reasoning, when the interactions between local ontologies are rather weak,

Table 1
The $E-\mathcal{SHIQ}$ constructors.

C	$C^{\mathcal{I}_i} \subseteq \Delta_i$
$\top_i$	$(\top_i)^{\mathcal{I}_i} = \Delta_i$
$\bot_i$	$(\bot_i)^{\mathcal{I}_i} = \emptyset$
	$R^{\mathcal{I}_i} \subseteq \Delta_i \times \Delta_i$, $(E_{ij})^{\mathcal{I}_{ij}} \subseteq \Delta_i \times \Delta_j$
	$(Inv(R))^{\mathcal{I}_i} = \{(x,y) \mid (y,x) \in R^{\mathcal{I}_i}\}$
$\neg C$	$(\neg C)^{\mathcal{I}_i} = \Delta_i \setminus C^{\mathcal{I}_i}$
$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}$
$(C \sqcup D)$	$(C \sqcup D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i}$
$(\exists R.C)$	$(\exists R.C)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \exists y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i}, y \in C^{\mathcal{I}_i}\}$
$(\forall R.C)$	$(\forall R.C)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \forall y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \rightarrow y \in C^{\mathcal{I}_i}\}$
$(\geq nR.C)$	$(\geq nR.C)^{\mathcal{I}_i} = \{x \in \Delta_i, \mid\mid y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i} \mid\mid \geq n\}$
$(\leq nR.C)$	$(\leq nR.C)^{\mathcal{I}_i} = \{x \in \Delta_i, \mid\mid y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i} \mid\mid \leq n\}$
$(\exists E_{ij}.G)$	$(\exists E_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \exists y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}}, y \in G^{\mathcal{I}_j}\}$
$(\forall E_{ij}.G)$	$(\forall E_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \forall y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}} \rightarrow y \in G^{\mathcal{I}_j}\}$
$(\geq nE_{ij}.G)$	$(\geq nE_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i, \mid\mid y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}} \wedge y \in G^{\mathcal{I}_j} \mid\mid \geq n\}$
$(\leq nE_{ij}.G)$	$(\leq nE_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i, \mid\mid y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}} \wedge y \in G^{\mathcal{I}_j} \mid\mid \leq n\}$

providing expressive inter-unit associations, but no subjectivity of specifications.

2.2.5. The $E-\mathcal{SHIQ}$ framework

The $E-\mathcal{SHIQ}$ distributed knowledge representation framework (Santipantakis & Vouros, 2014) provides a combination of the features available in DDL and $\mathcal{E}-Connections$. Specifically, $E-\mathcal{SHIQ}$:

- Provides subjective concept-to-concept correspondences between concept names in different units,
- Provides constructors for relating individuals in different units via link relations (representing domain-specific relations), as well as via subjective individual correspondences (representing subjective equalities between individuals in different units). Link relations may be further restricted via value and cardinality restrictions, and they can be hierarchically related with other link relations from the same unit.
- Provides the means for distribution of reasoning tasks among units, supporting them to reason with local knowledge, further propagating implications to others.

Any of the above mentioned modular representations formalisms may be used to support modular specification of knowledge and data w.r.t. their assumptions. In this article, aiming at a generic solution that (a) imposes no restriction to the potential domains of different units (thus, to the data of individual data sources), (b) respects the subjectivity of stakeholders' views on how data in sources are related, and (c) provides expressiveness for the representation of knowledge and a rich set of constructors for specifying associations among units, we have chosen $E-\mathcal{SHIQ}$ as the representation framework to be used.

Delving into the necessary formal details of $E-\mathcal{SHIQ}$, given the index I of units, let N_{C_i}, N_{R_i} and N_{O_i} be the sets of concept, role and individual names, respectively, for the ith unit $\mathcal{M}_i$. For some $R \in N_{R_i}$, $Inv(R)$ denotes the inverse role of R and $(N_{R_i} \cup \{Inv(R) \mid R \in N_{R_i}\})$ is the set of $\mathcal{SHIQ}$ i-roles, i.e. the roles of the ith unit $\mathcal{M}_i$. An i-role axiom is either a role inclusion axiom or a transitivity axiom. Let $\mathcal{R}_i$ be the set of i-role axioms.

The sets of i-concepts (denoted using the prefix "i: ") are inductively defined by the constructors shown in Table 1, where R and S are i-roles, and E_{ij} are link relations linking individuals from unit i to individuals in unit j.

Let i: C and i: D possibly complex concepts, and i: $C \sqsubseteq i$: D (or i: $C \sqsubseteq D$) a *general concept inclusion* (GCI) axiom. A finite set of GCI's is a TBox for the ontology unit $\mathcal{M}_i$ and it is denoted by $\mathcal{T}_i$.

Concepts' correspondences may be concept *onto* concept, or concept *into* concept: Let $C \in N_{C_i}$, $D \in N_{C_j}$ with $i \neq j \in I$. A concept *onto* (*into*) concept correspondence from unit $\mathcal{M}_i$ to unit $\mathcal{M}_j$ *from the subjective point of view of* j is of the form $i : C \xrightarrow{\sqsupseteq} j : D$ (corresp. $i : C \xrightarrow{\sqsubseteq} j : D$).

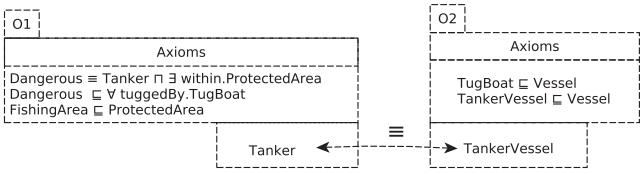

Fig. 1. A simple example of an $E-\mathcal{SHIQ}$ distributed knowledge base with two units.

An $E-\mathcal{SHIQ}$ *distributed knowledge base* $\Sigma = \langle \mathbf{T}, \mathbf{R}, \mathbf{C} \rangle$ is composed by the distributed TBox $\mathbf{T}$, the distributed RBox $\mathbf{R}$, and a tuple of sets of (subjective from the point of view of $\mathcal{M}_j$) correspondences $\mathbf{C} = (\mathbf{C}_{ij})_{i \neq j \in I}$ between units. A *distributed TBox* is a tuple of TBoxes $\mathbf{T} = (\mathcal{T}_i)_{i \in I}$, where each $\mathcal{T}_i$ is a finite set of i-concept inclusion axioms. A *distributed RBox* is a tuple of ij-property boxes $\mathbf{R} = (\mathcal{R}_{ij})_{i,j \in I}$, where each $\mathcal{R}_{ij}$ is a finite set of property inclusion axioms and transitivity axioms.

A *distributed ABox* (DAB) includes a tuple of local ABoxes $\mathcal{A}_i$ for each ontology i, sets $\mathcal{A}_{ij}, i \neq j$ with individual correspondences of the form $i{:}a \xrightarrow{=} j{:}b$ (that the unit $\mathcal{M}_j$ subjectively holds), and property assertions of the form (a, b): E_{ij}, where E_{ij} is a property for unit $\mathcal{M}_i$. Thus, individual correspondences, together with assertions concerning related and linked individuals, are made locally available to each unit.

A distributed knowledge base along with the distributed ABox, forms a *network of units*. Two units in such a network are connected if there is a non-empty set of associations between them. The direction of associations depends on the subjectiveness of concept correspondences and on the direction of link relations.

Specifications in units can be expressed in an at most $\mathcal{SHIQ}$ fragment of Description Logics and be de/serialized in standard OWL files. Link-relations and correspondences for both terminological and assertional specifications are included in enhanced C-OWL (Bouquet, Giunchiglia, Van Harmelen, Serafini, & Stuckenschmidt, 2003) files, that in addition to correspondences specify link-relation restrictions and assertions.

As an example of an $E-\mathcal{SHIQ}$ distributed knowledge base $\Sigma = \langle \mathbf{T}, \mathbf{R}, \mathbf{C} \rangle$, let us consider two ($I = \{1, 2\}$) units O_1 and O_2, depicted in Fig. 1).

The distributed RBox is:
$\mathcal{R} = ((R_i)_{i \in I}, (R_{ij})_{i \neq j \in I})$, $R_i = R_{ij} = \emptyset$, $i, j \in I$.
The distributed TBox $\mathbf{T}$ is:

$\mathcal{T}_1$ = {`Dangerous` ≡ `Tanker`⊓∃`within.ProtectedArea`,
`Dangerous` ⊑ ∀`tuggedBy.TugBoat`,
`FishingArea` ⊑ `ProtectedArea`},
$\mathcal{T}_2$ = {`TugBoat` ⊑ `Vessel`, `TankerVessel` ⊑ `Vessel`},
a tuple of sets of correspondences $\mathbf{C}$ is:
$\mathbf{C}_{21}$ = {`2:TankerVessel` $\xrightarrow{=}$ `1:Tanker`}
$\mathbf{C}_{12}$ = {`1:Tanker` $\xrightarrow{=}$ `2:TankerVessel`}
ABoxes $\mathcal{A}_1, \mathcal{A}_2$ and individual correspondences $\mathcal{A}_{21}$ are as follows:
$\mathcal{A}_2$ = {`v001:TankerVessel`},
$\mathcal{A}_1$ = {`s5742:Tanker`},
$\mathcal{A}_{21}$ = {`2:v001` $\xrightarrow{=}$ `1:s5742`}

The overall architecture of a peer responsible for a specific unit is shown in Fig. 2. The core component of such a peer is the local (peer specific) tableau algorithm (Santipantakis & Vouros, 2014). The connector component communicates messages to/from other peers, supporting the combination of local reasoning chunks (Santipantakis & Vouros, 2014).

Specifically, the $E-\mathcal{SHIQ}$ distributed reasoner implements a sound and complete tableau algorithm for combining local reasoning chunks corresponding to the individual units in a peer-to-peer fashion. In so doing it inherently supports the propaga-

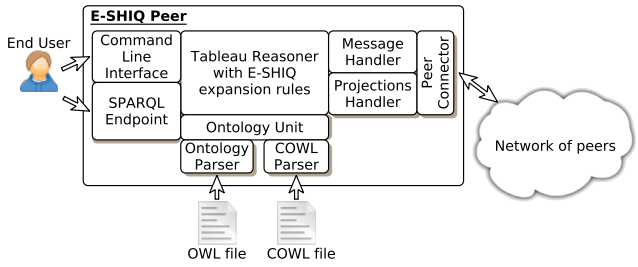

Fig. 2. The overall architecture of an $E-\mathcal{SHIQ}$ peer.

tion of subsumptions between the, so called, reasoning peers. The $E-\mathcal{SHIQ}$ reasoner implements a distributed tableau method for deciding concept satisfiability w.r.t. a knowledge base. Thus, the reasoner constructs a model (i.e. a directed graph called completion graph) that it satisfies all the constraints implied by the semantics of the (local and subjective) specifications in the knowledge base. Each peer is an active reasoning entity that exploits the knowledge in a specific ontology unit. However, this is done jointly with other peers: Reasoning in $E-\mathcal{SHIQ}$ involves combining peers' local reasoning chunks over ontology units. This is done by applying tableau expansion rules (Santipantakis & Vouros, 2014) that assure expansion and maintenance of the completion graph in each of the peers, and the proper maintenance of correspondences among individuals in different units, so as to retain the properties of the model constructed w.r.t. the semantics of specifications. No peer has a global, complete view of the overall completion graph. Each node and edge in the graph is labeled with a set of concepts and properties (i.e. roles or link relations), respectively. To combine local completion graphs, peers, according to the tableau algorithm for $E-\mathcal{SHIQ}$, use rules that project labels of individuals to their corresponding individuals in other units, updating their labels. This guarantees propagation of (subjective) constraints among peers and maintenance of a joined view of all the constraints that individuals in the model must satisfy. Further details on the reasoning mechanism are available in Santipantakis and Vouros (2014).

2.3. OBDA

State of the art OBDA frameworks provide a unified query-interface to support ontology-mediated access to data sources, and do not address the problem of retrieving and integrating data from multiple data sources. Widely used open source OBDA frameworks that support accessing the content of relational databases (RDB) and transforming them to virtual (read-only) RDF graphs are the D2RQ (Bizer & Seaborne, 2004) and Ontop (Calvanese et al., 2017). Other OBDA systems exist as well (e.g. Civili et al., 2013; Figueiredo et al., 2013; Pellegrini, Auer, Tochtermann, and Schaffert, 2009; Pinkel, Binnig, Kharlamov, and Haase, 2013; Poggi et al., 2008; Sequeda, Arenas, and Miranker, 2014; Szekely et al., 2014, but either they do not provide the code of their system, or do not provide state of the art solutions to accessing and processing data.

OBDA systems transform relational database sources to virtual read-only RDF graphs by utilizing RDB to RDF mappings (Arenas, Bertails, Prud'hommeaux, & Sequeda, 2012) (correspondences between RDB tables/columns and ontology concepts/properties), based on specific rules of a mapping language (e.g. D2RQ (Bizer & Seaborne, 2004) or the R2RML (Das, Sundara, & Cyganiak, 2012) mapping languages). An RDB to RDF transformation process may initially generate RDB to RDF mappings for relational data source. This can be done automatically by means of a default, auto-generated RDF schema (following simple rules of one-to-one correspondence from RDB to RDFs). This default RDF schema can be mapped to a reference ontology by identifying similarities between schema elements and the elements of the reference ontology. The objective is to use the corresponding ontology elements for describing data using an (as much as possible) agreed vocabulary with well-defined semantics. However, state-of-the-art approaches (Bagosi et al., 2014; Eisenberg & Kanza, 2012) allow the specification of mappings between data and ontologies, providing support for more intelligent retrieval tasks, further advancing the processing of data and the extraction of enriched information out of the data. Exploiting the RDB to RDF mappings, virtual RDF graphs are created: No persistent storage is required. These graphs are then accessible via corresponding SPARQL endpoints.

Towards delivering an implementation of OBDAIR, we have been experimenting with two existing ODBA frameworks, D2RQ (Bizer & Seaborne, 2004) and the Ontop (Kontchakov, Rezk, Rodríguez-Muro, Xiao, & Zakharyaschev, 2014). Detailed experimentation and lessons learned from these two OBDA frameworks has been presented in an early publication (Santipantakis, Kotis, & Vouros, 2015b), supporting our decision to use Ontop (Kontchakov et al., 2014). Ontop takes advantage of optimized SPARQL-to-SQL query rewriting techniques (Rodrıguez-Muro, Kontchakov, & Zakharyaschev, 2013), while it provides advanced flexibility of specifying RDB to RDF mappings.

Concerning RDB to RDF mappings there are three approaches: The local-as-view (LAV) approach that describes the data of a data source $\mathcal{S}$ by means of conjunctive queries against an ontology $\mathcal{O}$, the global-as-view (GAV) approach that defines concept and role specifications in $\mathcal{O}$ by means of conjunctive queries against the source data base $\mathcal{S}$, and finally, the global-local-as-view (GLAV), which states equivalences between queries against the source database and queries against the ontology. While LAV approaches are more robust to changes in the sources, GAV allow more efficient query processing techniques and GLAV do combine these two approaches.

Mappings in Ontop, following the GAV approach, are expressed either in a custom, easy to read mapping language, or in the RDB to RDF mapping language R2RML (Das et al., 2012). According to the GAV approach, a mapping is a set of rules of the form $\sigma(\mathbf{x}) \rightarrow E(\mathbf{x})$, where the *source* part of the mapping $\sigma(\mathbf{x})$ is a conjunction of atoms with database relations or views and a filter, which is a Boolean combination of built-in predicates (i.e. an SQL query over $\mathcal{S}$). The *target* of the mapping E is a functional expression that creates new assertions in the ontology.

For instance, an Ontop mapping in the maritime domain that recognizes the tanker vessels as individuals of the concept `Tanker`, assuming that vessel types are available in more than one table, i.e. in the tables `R1`, `R2`, is as shown in Fig. 3. This example uses regular expressions for the values of `typeStr` and `type_` to select the appropriate tankers from the data source.

Concerning the semantics of the virtual ABox constructed by means of data from the source, let $\Delta\Sigma = \langle \mathcal{O}, \mathcal{S}, \mathcal{A} \rangle$ be an OBDA system, where $\mathcal{O}$ is an ontology, $\mathcal{S}$ is a data source, $\mathcal{A}$ is a mapping between the ontology and the data source, and δ a database instance w.r.t. $\mathcal{S}$. We say that an interpretation $\mathcal{I}$ is a model of $\Delta\Sigma$ w.r.t. δ if it satisfies $\mathcal{O}$ and the data in δ over the mapping $\mathcal{A}$, i.e. $\mathcal{I} \vDash \mathcal{O}$ and $\mathcal{I} \vDash_{\mathcal{A}} \delta$, where the definition of $\vDash_{\mathcal{A}}$ depends on the mapping approach. Specifically, for the GAV approach we say that $\mathcal{I} \vDash_{\mathcal{A}} \delta$ if there is an ABox $\mathcal{A}_{Box}$ s.t. $\mathcal{A}_{\delta} \subseteq \mathcal{A}_{Box}$ and $\mathcal{I} \vDash \mathcal{A}_{Box}$, where $\mathcal{A}_{\delta}$ is the ABox produced from applying $\mathcal{A}$ on δ.

3. The OBDAIR integrated framework

This section presents the Ontology-based Distributed framework for Accessing, Integrating and Reasoning (OBDAIR) with data from disparate and heterogeneous data sources. OBDAIR is engineered using five main building blocks:

Source:

```
SELECT imo, vessel_name as vName, typeStr as vType
FROM R1
WHERE (typeStr LIKE '%TANKER%') OR (typeStr LIKE
'%BUNKERING%') OR (typeStr = 'CHEMICAL') OR (typeStr
= 'CONTAINER') OR (typeStr = 'CRUDE') OR (typeStr =
'LNG') OR (typeStr = 'LPG')
UNION
SELECT imo_no as imo, name as vName, type_ as vType
FROM R2
WHERE (type_ LIKE '%TANKER%') OR (type_ LIKE
'%BUNKERING%') OR (type_ = 'CHEMICAL') OR (type_ =
'CONTAINER') OR (type_ = 'CRUDE') OR (type_ = 'LNG')
OR (type_ = 'LPG')
```

Target:

```
:vessel{imo} a :Tanker ; :hasName {vName} ; :typeOf
{vType}.
```

Fig. 3. OBDA mapping example.

- Multiple ontology units for the description of data in disparate data sources,
- Correspondences between concepts and individuals of ontology units, so as to couple ontology units[1] into a distributed knowledge base and integrate data into a coherent view,
- RDB to RDF transformations (mappings) for ontology-based data access,
- Ontology-mediated distributed data access to multiple sources, query-answering and reasoning components,
- Data processing close to the data sources (i.e. prior to populating the ontology) for extracting information out of raw data.

3.1. Overall architecture

The overall architecture of the proposed framework is shown in Fig. 4. As already stated, the key idea is to support distribution of knowledge, information and tasks in multiple peers with interlinked ontology units. OBDAIR architecture consists of four layers and intermediate components necessary to orchestrate tasks in the layers involved, also coordinating data flows between layers.

Specifically, OBDAIR consists of the OBDA framework which is in charge for retrieving information from data sources, to populate the modular ontology framework in the second layer. The OBDA connectors are configured by the OBDA mappings, using the mappings specification language of the OBDA framework. The top layer is a SPARQL Service Framework, which is responsible to direct user queries towards the modular ontology framework and serve with result sets. This architecture provides an additional layer for processing data, which is interposed between the OBDA framework and the modular ontology framework. This layer incorporates methods for data conversion, information extraction and integration. While integration tasks are necessary for associating entities and coupling information in different units, extraction aims to identify information that is necessary for the efficient performance of reasoning tasks. Thus, information extraction and integration tasks must always precede any reasoning task, so as to support efficiency, preserve coherence of specifications in the distributed knowledge base, and thus completeness and soundness of the reasoning tasks. Thus, the functionality of this layer is highly dependent on the modular ontology framework used, to the semantics of the inter-unit associations and to the ontology specifications involved in any reasoning task. Information extraction and integration are presented in Sections 3.3 and 3.4.

[1] Any type of constructs from those mentioned in Section 2.2 can be used for specifying inter-unit associations.

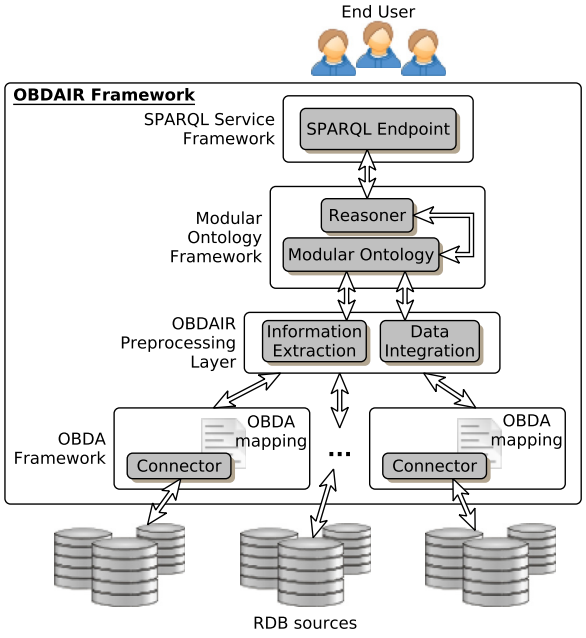

Fig. 4. The OBDAIR framework overall architecture.

The modular ontology framework is responsible for the reasoning tasks and the detection of complex events. In the implemented instance of OBDAIR, we have employed $E - \mathcal{SHIQ}$, which enables monotonic reasoning in $\mathcal{SHIQ}$ fragment of Description Logics (under open world semantics), and above OWL2QL.

Finally, the SPARQL service framework provides an endpoint for query-answering. A selected unit may act as the interface between the endpoint and the reasoner. Alternatively, there may exist one endpoint per ontology unit, supporting federated queries. The OBDAIR framework directs the queries from the SPARQL endpoints to the model inferred by the distributed reasoner. Thus, the query result set includes information derived from the OBDAIR preprocessing layer and the distributed reasoner.

OBDAIR configuration includes a set of options that specify how the retrieved data should be exploited. Specifically, data in sources can be distinguished to static or dynamic, where the former can be retrieved only once at the initialization phase and will change rarely or quite infrequently, while the latter are retrieved on predefined time periods and have an expiration time. This is further discussed in Section 3.2.

Concerning extendability, additional data sources can be connected to the framework, at any time, the information extraction and link discovery components can be expanded with more rules, and additional peers/units can be employed w.r.t. the data sources and the modular ontology framework used. Additional peers allow further distribution of data and knowledge, further distribution of data retrieval tasks, and scalability on integration and reasoning tasks. However, functional details such as balancing the workload among peers, are considered as part of the modular ontology framework, and they are not addressed in this article.

The extensibility of OBDAIR architecture highlights another interesting perspective of this work: the construction of hierarchical OBDAIR systems, where OBDAIR at higher levels in the hierarchy extract information and implications inferred from lower level OBDAIR systems. The connections between OBDAIR instantiations can be established either in the SPARQL Service Framework layer (i.e. OBDAIRs communicate with SPARQL queries and result sets),

or at the Modular Ontology Framework layer, w.r.t. the functionalities, compatibility and requirements of the corresponding reasoners. Each option has certain benefits and requirements. Specifically, the SPARQL service connectivity allows independence of reasoning tasks on distinct OBDAIR instantiations, but it may result to expensive join operations, while it cannot compute any further inferences resulting from the joined result, given that the SPARQL service framework has no reasoning capabilities. On the other hand, connectivity at the Modular Ontology Framework layer tackles the problem of inferring knowledge over the joined set from different OBDAIR instantiations, yet it should support propagation of subsumptions and directionality of associations between units.

3.2. Accessing and processing data

Beyond the existence of disparate data sources and the heterogeneity of data, the rate at which data are updated or created, plays an important role in a modeling and processing strategy: Static data, which do not change frequently, are usually archival data such as the name and type of ports or vessels, whereas dynamic data, which change or are being updated quite frequently, usually concern positional information of moving objects, weather data, or measurements reported by sensors.

As it can be easily seen, the distinction between static and dynamic data is important to the design of the OBDAIR architecture: Static data can be retrieved, processed, and asserted in ontologies only at system initialization; while the amount of dynamic data in the virtual ABox's need to be reduced so as to reduce the required computational resources for the performance of integration and reasoning tasks, together with reducing the need for large ABox's updates. The objective here is to make the necessary assertions, at the level of abstraction that is required towards achieving the reasoning goals, only.

Generally, the data retrieval process can be seen as a time dependent function $\mathbf{d} = f(q, T)$, where for a given time instant or period T and a query q, the function returns the results $\mathbf{d}$, which may result from processing raw data in the sources. Given a query q for static data the time variable T is theoretically infinite (i.e. peers can retrieve data only once), while for dynamic data the values of T depends on the frequency of data updates in the sources and may vary depending on the type of data. In this work we consider that all dynamic data are updated at specific time intervals and are retrieved at the end of that interval, although in the most general case this is not necessary. The duration of this interval is a system parameter that must be configured appropriately. OBDAIR does not have the objective to process streaming data in real-time, although it does process dynamically changing data stored in a relational database (much like a micro-batching approach).

Beyond retrieving data, another key issue in our work is the processing of data prior to populating the ontology units and while they are being retrieved, i.e. close to the data sources, following the in-situ processing paradigm: Our main goal is to save reasoning time by extracting only the assertions that peers need for the detection of complex event patterns, although in this case the data retrieval time may increase due to the complexity of queries. This is in contrast to populating units with raw data retrieved from the sources, which in OBDAIR is always an alternative approach. Not surprisingly, data processing functions for extracting information (ABox assertions) out of data are distinguished into two main categories: (a) functions for the population of ontological concepts, and (b) functions for the detection of domain-specific relations between instances. To achieve a good balance between data retrieval and data processing/reasoning tasks efficiency, we propose the use of *triple templates* presented in Section 3.3.

Furthermore, computing and asserting relations between instances can be computationally demanding, especially when spatiotemporal relations among entities need to be detected. For instance, in the maritime domain we have to compute such relations between vessels, and between vessels and sea regions. In the general case the computation of relations between entities may be demanding if multiple features, specific types of constraints and various types of interrelations among entities must be considered. Currently, we have implemented a component for computing specific relations as discussed in Section 3.5. In further improvements of the OBDAIR framework, we plan to extend the preprocessing layer to integrate link discovery frameworks such as LIMES (Ngomo & Auer, 2011), SILK (Isele & Bizer, 2013), or ORCHID (Ngomo, 2013).

Subsequent subsections elaborate on the most important functionality of the OBDAIR architecture.

3.3. Extracting information by means of triple templates

For the recognition of instances' types and relations, sufficient data about these instances must be retrieved from the sources. This may result in a large number of assertions in ontology units, while it may increase the reasoning time for realizing entities as concept instances and for recognizing specific relations between them. On the contrary, extracting information out of raw data may require complex queries to the sources. To address this issue and aiming to achieve efficient data retrieval while supporting efficient reasoning with data, we have implemented a wrapper on the SPARQL query engine that allows the specification of *triple templates* for the automatic construction of triples from query results.

Each triple template is signalled by the keyword `ASSERT`, using variables specified in the SPARQL query. When the result set for the query is served to the client, each variable in the `ASSERT` expressions is instantiated to a specific value, forming additional triples to the query results. Formally, let $\mathbf{V}$ be a set of variables $\{?v_1, ?v_2, \ldots\}$, disjoint from the set of terms $\mathbf{N}$ available in a data source D. An RDF *triple* is a *tuple*(s, p, o) in $\mathbf{N} \times \mathbf{N} \times \mathbf{N}$ (without considering blank nodes and all literals assumed in $\mathbf{N}$) and a *triple pattern* is in $(\mathbf{N} \cup \mathbf{V}) \times (\mathbf{N} \cup \mathbf{V}) \times (\mathbf{N} \cup \mathbf{V})$. A set of triple patterns forms a graph pattern. An instantiation of a triple (or graph) pattern G results from a mapping $\mu \in \mathbf{M}$, $\mu : V_G \rightarrow \mathbf{N}$, substituting all variables in the pattern G (denoted by V_G, $V_G \subseteq \mathbf{V}$) with specific terms in $\mathbf{N}$.

A triple template query is an expression in the form:

```
ASSERT R SELECT V WHERE P
```

where R and P are graph patterns involving variables in V, such that $V_R \subseteq V_P$. The result of such a query includes all triples instantiating P according to a mapping $\mu \in \mathbf{M}$, $\mu : V_P \rightarrow \mathbf{N}$, as well as additional triples instantiating R with respect to that mapping.

For example, the safety regulations for vessels of International Maritime Organization[2] define that a high-speed craft is capable of a maximum speed, in metres per second (m/s), equal to or exceeding 3.7 times the one-sixth power of the volume of displacement (in m^3) corresponding to the design waterline. This specification can be used to realize `HighSpeed` individuals in the corresponding ontology. Although such an instance is hard to be realized using a DL reasoner, or it requires additional processing of data watched from the sources, it can be easily specified using a triple template as follows (where $3.7^6 = 2565.73$):

```
ASSERT ?x a :HighSpeed
SELECT ?x ?m ?d
WHERE{ ?x :maxSpeed ?m ; :draught ?d .
FILTER(?m*?m*?m*?m*?m*?m>2565.73*?d).}
Result set:
:FD_ATHINA :maxSpeed ''17.58''.
:FD_ATHINA :draught ''1.8''.
Asserted triple:
:FD_ATHINA a :HighSpeed
```

[2] http://www.imo.org/OurWork/Safety/Regulations/Pages/Default.aspx.

where 17.58 m/s is approximately 34 knots.

Although the ASSERT share similar features to the standard SPARQL 1.1[3] CONSTRUCT, it has significant differences. The SPARQL 1.1 CONSTRUCT substitutes the variables in a given graph pattern with values resulting from satisfying the WHERE part of the query. On the other hand, ASSERT, applies SELECT and appends additional triples in the result set. On a more technical point of view, ASSERT addresses the limitation of OBDA query engines that do not allow CONSTRUCT queries from clients, and at the same time enables additional triples to be appended in the result set.

Some of the most important consequences of using triple templates are as follows:

- The functions used in an OBDA mapping must be supported by the RDBMS in a data source.
- The SQL functions and SPARQL operators in the OBDA mapping may complicate the translation and result to poor performance. As a rule-of-thumb, we prefer as simple as possible OBDA mappings.
- The triple template implies that the additional triples will be constructed in the client side. This means that there is no additional overhead for the OBDA, since the task is in the responsibility of the peer. This can lead to better distribution/parallelisation of tasks.

3.4. Data integration

Data integration concerns (a) the transformation of data residing in different sources to common formats, and (b) the detection of equalities and other relations between instances of ontology concepts.

OBDA mapping languages allow the use of functions and operators to convert data to specific formats. For example, geometries and positions may be required in the Well Known Text (WKT) format, however data sources may provide location information using longitude (lon) and latitude (lat). In this case the WKT for the position of a vessel can be produced by the following Ontop mapping:

Source:
```
SELECT 'POINT(' || lon || ' ' || lat || ')' as wkt, lon,
lat FROM R1
```
Target:
```
:SpObj{lat}_{lon} a :SpatialObject ;
:hasGeometry {wkt} .
```

The position of each SpatialObject instance is assigned as a WKT value to the data property :hasGeometry. The mapping in this example constructs the WKT using text concatenation in the source part of the mapping.

As another example, geometries referenced in different geodetic coordinate systems can be transformed to a specific (defined by the application) coordinate system using the st_transform(geom, srid) function in the source part of the mapping. Such a mapping would be:

Source:
```
SELECT st_asText(st_transform(geom,4326)) as wkt, gid
FROM R2
```
Target:
```
:SpObj{gid} a :SpatialObject ;
:hasGeometry {wkt} .
```

where 4326 is the Spatial Reference System Identifier (SRID) for WGS84. The data conversion in the above examples resides on the functions available in the RDBMS data source. Thus, some OBDA mappings can be satisfied only by certain data sources.

[3] http://www.w3.org/TR/sparql11-query/.

Data integration also concerns linking entities. This may concern computing equalities among entities, or specific types of relations between them. The computation of relations is presented in Section 3.5. Equalities between concept instances are distinguished to equalities between instances in a specific unit, and equalities between instances in different units. The former are necessary when a peer retrieves data from different sources and has to unify entities and consolidate data about them. The latter results to instances correspondences between units w.r.t. the modular ontology framework employed, and in the general case requires the collaboration of peers operating in the network of ontology units. To detect equalities, peers consult patterns of data properties provided by domain experts. Corresponding individuals must satisfy one of these patterns. To reduce the comparisons between individuals, the Data Integration component also exploits the concept-to-concept correspondences specified in the distributed knowledge base.

The matching method is described as follows: The process of instance matching is triggered when new instances i: x are asserted in a unit $\mathcal{M}_i$ as instances of a concept i: C. If assertions have been made for data properties involved to any pattern $p \in \mathcal{P}$, and there is a concept-to-concept correspondence of the form j : $D \xrightarrow{\#} i : C$, $\# \in \{\equiv, \sqsubseteq, \sqsupseteq\}$, for i: C with any of the neighbor units $\mathcal{M}_j$, the process will compute individual equality correspondences with individuals in $\mathcal{M}_j$ that are instances of corresponding concepts j: D. The originating peer i (corresponding to unit $\mathcal{M}_i$) translates the data properties to the corresponding data properties of $\mathcal{M}_j$, as well as their values if necessary, and requests all the individuals y, that satisfy p (i.e. they have the specific values for the specified data properties) and are instances of j: D, thus pruning the search to the instances of concepts corresponding to i: C. The peer j will report the URIs of those instances of j: D matching p. For each URI received by the originating peer, an individual correspondence to the corresponding instance of $\mathcal{M}_j$ is specified. In case of equivalence or *onto* correspondences, and in case no individual of peer j matches the pattern p, peer i must inject new individuals in unit $\mathcal{O}_j$, to preserve the semantics of correspondences.

It must be noticed that for the case of Modular Ontology frameworks that support subjective correspondences, peers in the general case may need to reach agreements on their instance correspondences (Vouros, 2014). This process however is not in the scope of this work. Actually, OBDAIR allows establishing bi-directional correspondences, i.e. correspondences between units $\mathcal{M}_i$ and $\mathcal{M}_j$ computed for ontology unit $\mathcal{M}_i$ automatically are set for the unit $\mathcal{M}_j$, as well. This is with no loss of generality since it is assumed that all peers know the same set of patterns $\mathcal{P}$ (implying that peers a priori agree to the individual correspondences detected). However, it must be pointed out that still peers are allowed to have different patterns to compute subjective equalities.

This method for computing individual correspondences benefits from the distribution of data to peers, while preserving the subjectivity of specifications among units, if it is necessary to do so.

3.5. Computation of instances' relations

Although triple templates support peers to extract information out of raw data and to make the appropriate assertions in their ABoxes, the assertion of relations that require more elaborated computations in an n-ary space, considering n instances' features, need to be supported. An example of such relations are the spatiotemporal relations among entities, which concern at least 3 features of the instances.

The main -and widely used idea- is to first organize the n-dimensional space into a predefined number of disjoint and adjacent segments: For n = 2, these are also referred as rectangles or tiles shaping a grid. Each individual for a specific time instant is "placed" in the corresponding segment of the n-dimensional space.

Individuals corresponding to the same segment form a *block*. Relations are computed only between individuals in the same block. This can be done in parallel for each unit and for the different segments. Finally, possible relations between individuals that fall near the borders of adjacent segments must be computed. The proposed framework benefits from the distribution of these computations, and from the fact that OBDA mappings, queries and ontology specifications are kept as simple as possible.

Specifically, the management of the computation of spatiotemporal relations is performed by a single and independent process in each peer. The process is initialized by a settings file which allows the construction of the grid, defining the space segments and the default SRID to be used for the geometries (currently using 4326).

The settings file also specifies the following:

1. The maximum number of concurrent threads we allow for the computation of spatiotemporal relations (currently we compute two types of relations: "within" and "nearby"),
2. The name of the term to be used for asserting nearby relations among entities (the default is `geosparql:nearby`),
3. The name of the relation that will be used for assertions expressing within spatial relations (the default is `geosparql:within`),
4. The name of the property that specifies geometries. The default term in our implementation is `:hasGeometry`,
5. The distance threshold *D*th for determining a nearby relation. The default value is 1 km, e.g. entities within the range of 1 km are considered to be related.

Given the above settings, each update of dynamic data triggered by a peer results to the retrieval of new triples for the corresponding ontology unit. The process scans the newly retrieved triples for any triples identifying a geometry. Then, it assigns each block to a worker, and distributes the new triples involving geometries to the workers: Each geometry that overlaps with a segment of the grid, is assigned to the corresponding worker. Processes assigned to workers are executed in parallel, and each one is responsible for computing "nearby" and "within" relations between geometries of entities in the domain.

Specifically,[4] two individuals *x, y* s.t. *:hasGeometry(x,gm1)* and *:hasGeometry(y,gm2)*, where geometries *gm1,gm2*, are considered to be *nearby* if their Euclidean distance is at most equal to the distance threshold *D*th. Similarly, an individual *x* s.t. *:hasGeometry(x,gm1)* is said to be "within" an individual *y* s.t. *:hasGeometry(y,gm2)*, if every point of *gm1* is within *gm2*, i.e. $gm1 \cap gm2 = gm1$. Since we are dealing with moving entities, a "nearby" or "within" spatial relation holds, only if the temporal intervals associated to the geometries, overlap.

The system uses the open source JTS topology suite[5] to deserialize the geometries retrieved from the data sources and to compute the spatial relations among entities w.r.t. the time period associated to their positions.

4. OBDAIR in the maritime domain

The proposed framework was initially motivated by, developed for, and tested in, the context of AMINESS (Analysis of Marine INformation for Environmentally Safe Shipping) project,[6] aiming to deliver integrated data at higher levels of abstraction towards supporting recognition of complex marine-related events.

Maritime navigation technology can automatically provide real-time information from sailing vessels. The Automatic Identification System (AIS) (AIS, 2016) is a tracking system for identifying and locating vessels at sea through data exchange, intended to assist in monitoring vessel movement and maritime situation awareness. AIS raw tracking data offers a wealth of information including unique identification of vessels, their position, course, and speed, while AIS information is continuously emitted from over 400 thousand ships worldwide (MarineTraffic, 2016).

To exploit AIS messages in the context of the AMINESS project, a trajectory detection component that consumes a positional stream of AIS messages from a large fleet and tracks major changes along each vessel's movement (Patroumpas et al., 2015) has been provided. This identifies "critical points" en route, indicating important changes like a stop, a sudden turn, or slow motion of a ship, producing a synopsis of each vessel trajectory. Critical points computed update the RDB named Hermes.

4.1. Modeling data, knowledge and correspondences

The data sources available in the application domain are the following:

(a) Hermes data source (a PostGIS/PostgreSQL, operational at the University of Piraeus), storing data related to vessels' trajectories' synopses identified via the computation of vessels' movement critical points, and
(b) Aminess data source (a PostGIS/PostgreSQL, operational at the University of the Aegean), providing data gathered from different sources with no prior processing towards their integration or fusion: It includes data from both maritime (vessels' and ports' details, geographical areas, accidents at sea, depths and coastlines of specific protected areas, fishing areas, etc.) and weather (detailed real weather conditions at specific time and space) data sources.

Data sources provide both static/archival and dynamic data: Hermes provides dynamic data about vessels' movement, while the Aminess data source provides both static and dynamic data. Given the data provided by the sources, we devised four distinct ontologies as if it were to model four distinct data sources:

(a) The "Hermes" ontology unit specifying critical points only (as low level events) and trajectories,
(b) The "Aminess static" ontology unit specifying details on vessels static characteristics, ports and areas of specific interest (fishing areas, or areas of rich bioversity, protected areas, etc.),
(c) The "Aminess dynamic" ontology unit specifying real weather conditions, and
(d) The "Events" ontology unit specifying maritime low-level (simple), and high-level (complex) events.

The "Events" ontology unit was engineered for representing patterns of complex maritime events: These are specified by means of concepts, relating simple events concerning vessels' movement, static data about vessels, sea areas, ports as well as spatiotemporal relations and dynamic data concerning weather conditions. High-level events related to vessels are recognized as instances of these ontology concepts.

In the following subsections we provide a short description of the most important ontological specifications for each of these units. It must be pointed out that not all concepts and properties are specified, while only examples of correspondences between units and examples of RDB-to-RDF mappings are provided. Complete specifications can be found at the following web pages:

- Ontologies (in OWL) http://ai-group.ds.unipi.gr/ontologies/owl.zip
- Ontop mappings (in *native Ontop mapping language*): http://ai-group.ds.unipi.gr/ontologies/obda.zip

[4] These specifications use *:hasGeometry(y,gm2)* instead of *(y,gm2):hasGeometry* used in examples.
[5] https://sourceforge.net/projects/jts-topo-suite/.
[6] http://aminess.eu/.

- $E-\mathcal{SHIQ}$ correspondences (in COWL): http://ai-group.ds.unipi.gr/ontologies/cowl.zip.

4.1.1. Modeling Hermes data

Vessels are objects with a unique identifier i.e. a Maritime Mobile Service Identity (MMSI), associated with trajectories.

The ontology specifications (in N3 format) related to a Vessel definition are shown below:

```
:vesselId rdfs:Domain :Vessel ;
rdfs:Range xsd:integer .
:vesselToTrip rdfs:subPropertyOf :associatedWith ;
rdfs:Domain :Vessel ; rdfs:Range :Trip .
```

The concept Vessel (hermes:Vessel) is mapped to the "imisObjects" table of Hermes RDB using the following RDB-to-RDF mapping:

```
Source:
SELECT obj_id FROM imis_obj
Target:
ontology:imis_obj_{obj_id} a ontology:Vessel.
```

Critical points of vessels at sea are simple/low-level events derived from the status of vessels. Such points are described by their duration (e.g. the duration of a change in direction or speed) and location (actually, the centroid longitude and latitude of an area).[7]

A trajectory (`hermes:Trajectory`) is a geometry (`sf:Geometry`) as represented in the Simplified Features Geometry ontology,[8] associated with a vessel `hermes:Vessel`), a port of origin and a destination port.

The concept Trajectory (`hermes:Trajectory`) is mapped to the 'imis_traj' table of the Hermes RDB. The RDB-to-RDF mapping specified in the Hermes ontology unit is:

```
Source:
SELECT obj_id, traj_id, origin, destination,
st_asText(geom) AS geomWKT FROM imis_traj
Target:
ontology:imis_traj{obj_id}_{traj_id} a
ontology:Trajectory ; ontology:trajectoryId {traj_id}
; ontology:trajectoryOriginPort {origin} ;
ontology:trajectoryDestinationPort {destination} ;
ontology:trajectoryGeom {geomWKT} ;
ontology:trajectoryToVessel
ontology:imis_obj_{obj_id} .
```

4.1.2. Modeling Aminess static data

The aim of this ontology unit is to model the static information provided by the Aminess data source. A vessel (aminessStatic:Vessel) is a craft (aminessStatic:Craft), with properties concerning the vessels' International Maritime Organisation number (IMO), Maritime Mobile Service Identity (MMSI), year of construction, flag and type. Several vessel types are defined as subconcepts of vessel, e.g. `aminessStatic:Fishing`, `aminessStatic:OilTanker`, `aminessStatic:HighSpeed`, `aminessStatic:BulkCarrier`, `aminessStatic:Yacht`.

Ports and Seaports are represented in this ontology unit as `aminessStatic:WPI` (World Port Index) and `aminessStatic:Seaport` respectively, the latter being a subconcept of the former.

Fishing areas (`aminessStatic:FishRegion`) are specified to model the related data from the Aminess RDB (table "fish_regions").

An example of an RDB-to-RDF mapping related to this ontology unit is as follows:

```
Source:
SELECT id, name, st_asText(st_centroid(geom)) AS geomWKT
FROM seaports2
WHERE st_makeenvelope(16.18, 33.07, 32.42,
41.86) st_centroid(geom)
Target:
seaport{id} a :SeaPort ; :seaPortGid {id} ; :seaPortName
{name} ; :hasGeometry {geomWKT} .
```

The specific mapping is using spatial functions (st_asText, st_makeenvelope and st_centroid) in order to convert types of data and also to filter the available data w.r.t. spatial restrictions (e.g. seaports located within the Greek region).

4.1.3. Modeling Aminess dynamic data

The aim of the "aminessDynamic" (prefix `dyn:`) ontology unit is to model the dynamic information available, such as AIS messages and weather conditions. Although the current system implementation does not exploit raw AIS data, we have modelled this information using the concept `dyn:maritimeAISdata`. This ontology unit is mainly used to model information on weather conditions (`dyn:MarineWeatherData`) for specific time and day (`dyn:weatherConditionDate`, `dyn:weatherConditionTime`) and for a specific location (`dyn:weatherConditionLat`, `dyn:weatherConditionLong`). Doing so, weather conditions can be related to specific vessels at a certain point in time and space.

Examples of RDB-to-RDF mappings related to this unit are as follows:

```
Source:
SELECT id, date_stamp, ''time'', max_temperature,
min_temperature, wind_speed_kmph, wind_dir_degree,
wind_direction, visibility, cloud_cover, swell_height_m,
latitude, longitude, 'POINT(' || longitude || ' ' ||
latitude || ')' as geom
FROM weather_data
WHERE st_makeenvelope(16.18, 33.07, 32.42,
41.86) (st_setsrid(st_makePoint(longitude,
latitude),4326))
Target:
:weather_data_{id} a :MarineWeatherData;
:weatherConditionDate {date_stamp};
:weatherConditionTime {time}; :weatherMinTemperature
{min_temperature}; :weatherMaxTemperature
{max_temperature}; :weatherConditionWindSpeedKm
{wind_speed_kmph}; :weatherConditionWindDirection
{wind_dir_degree}; :weatherConditionVisibility
{visibility}; :weatherConditionSwellHeight
{swell_height_m}; :weatherCloudCover {cloud_cover};
:weatherConditionLat {latitude}; :weatherConditionLong
{longitude}; :hasGeometry {geom} .
```

This mapping is using spatial functions (st_makeenvelope, st_setsrid and st_makePoint) to filter the available data in terms of spatial restrictions (i.e. get the weather data reported for coordinates in the Greek region).

Weather conditions are used for the specification of complex events in the Events ontology. This is done by means of $E-\mathcal{SHIQ}$ concept-to-concept correspondences, such as:

$$\texttt{events:BadWeatherData} \overset{\equiv}{\rightarrow} \texttt{dyn:BadWeatherData}$$[9]

4.1.4. Representing high-level events

The aim of the "Events" ontology unit is to specify patterns of complex maritime events in order to support the automated detection of events by means of the $E-\mathcal{SHIQ}$ reasoning capabilities. According to the following definition, an event is a spatiotemporal entity that is associated to at least one vessel.

[7] A critical point may be related to a time interval and a spatial area, given that such a point summarises multiple positions of a vessel reported by means of AIS messages in that time interval.

[8] http://www.opengis.net/ont/sf#.

[9] Equivalence correspondences are specified by means of onto and into concepts' correspondences.

$$\texttt{Event} \sqsubseteq \exists\, \texttt{locatedAt.Point} \sqcap \texttt{TemporalEntity} \sqcap \exists\, \texttt{eventAssociatedWithVessel.Vessel}$$

Concerning movement patterns, events specified are about:

- moving vessels (concept `events:VesselInKinesis`). More specifically, we have defined specific classes of events defining patterns of normal, abnormal-low and abnormal-high speed: `events:VesselInAbnormalSpeed`, `events:VesselInHighSpeed`, `events:VesselInLowSpeed`, `events:VesselInNormalSpeed`), and `events:VesselInTurn`
- vessels that have lost communication with the AIS base: This is represented by the concept `events:VesselInLostConununication`, and
- vessels that are not moving: Types of these events are specified by concepts `events:VesselInPort`, `events:VesselNotInPort` and `events:VesselInStop`.

High-level events are specified in the ontology unit as either area events (`events:AreaEvents`) associated with a specific type of area, or vessel events (`events:VesselEvents`), concerning a vessel (`events:Vessel`) located at some geographical point (`sf:Point`) at a specific time instant (`events:TimeObjects`):

Important to the specification of complex events is also the modeling of specific areas of interest. For instance, `events:FishingArea`, `events:ForbiddenArea`, `events:ShallowWatersArea` and `events:UnsafePort`, are examples of concepts that are used to indicate the spatial occurrence of an event. Thus, the event pattern `events:FishingAreaViolation` is defined as a vessel event involving a non-fishing vessel within a fishing area. This example definition is shown below:

$$\texttt{FishingAreaViolation} \equiv \texttt{VesselWithinRestrictedArea} \sqcap \exists\, \texttt{eventAssociatedWithVessel.}(\neg\texttt{Fishing} \sqcap \exists\, \texttt{within.FishingArea}) \sqcap \texttt{Event}$$

The concept `events:VesselWithinUnsafePort` is defined as a vessel within a restricted area (`events:VesselWithinRestrictedArea`) which is an unsafe port (`events:UnsafePort`):

$$\texttt{VesselWithinUnsafePort} \equiv \texttt{VesselWithinRestrictedArea} \sqcap \exists\, \texttt{withinArea.UnsafePort}$$

An `events:UnsafePort` is a seaport (`events:SeaPort`) and a restricted area (`events:RestrictedArea`), where the seaport is nearby some location with bad weather (`events:BadWeatherData`):

$$\texttt{UnsafePort} \equiv \exists\, \texttt{nearby.BadWeatherData} \sqcap \texttt{RestrictedArea} \sqcap \texttt{SeaPort}$$

Example of RDB-to-RDF mappings for this unit are as follows:

Source:
```
SELECT mmsi FROM critical_points
```
Target:
```
:vessel{mmsi} a :Vessel ; :vesselId {mmsi}.
```

Source:
```
SELECT mmsi, t1, t2, lon, lat, flag, speed, heading,
'POINT(' || lon || ' ' || lat || ')' as geom FROM
critical_points
```
Target:
```
:event{mmsi}_{t1}_{t2}_{flag} a :SimpleEvent ;
:EventStartEpoch {t1}^^xsd:long ; :EventEndEpoch
{t2}^^xsd:long ; :movingObjectEventSpeed {speed} ;
:movingObjectEventHeading {heading} ;
:movingObjectEventFlag {flag} ; :associatedWithArea
:SpObject{lon}_{lat} ; :eventAssociatedWithVessel
:vessel{mmsi} . :SpObject{lon}_{lat} :hasGeometry
{geom}^^xsd:string.
```

Examples of $E-\mathcal{SHIQ}$ correspondences related to this unit are shown below:

$$\texttt{aminessStaticData:Vessel} \overset{\equiv}{\rightarrow} \texttt{events:Vessel}$$
$$\texttt{hermes:VesselEvent} \overset{\equiv}{\rightarrow} \texttt{events:Event}$$
$$\texttt{dyn:BadWeatherData} \overset{\equiv}{\rightarrow} \texttt{events:BadWeatherData}$$

4.2. Reasoning with maritime data

As already stated, static data are retrieved only during system's initiation. Dynamic data are updated every T time points: An update is triggered by each $E-\mathcal{SHIQ}$ peer, requesting dynamic data from the data source(s), updating its own ABox. The update period can be configured appropriately according to specific needs. The goal should be to recognise all events related to the data within the time period of T time instants, while keeping T as low as possible: In other words, it must recognise all occurring events as efficiently as possible, before the next update.

To recognise events, spatiotemporal relations concerning (a) the proximity between vessels and other entities/areas (represented by the relation `nearBy`) and (b) the inclusion of a vessel within an area (represented by the relation `within`), must be known in advance. Data are processed to extract this kind of information, as already described in previous sections.

The following example demonstrates the overall process of combining and reasoning with data from different sources:

Example 1 (Vessel within protected area)**.** We consider a two-peers setting (Fig. 5(f)), with $E-\mathcal{SHIQ}$ units $\mathcal{O}_1$ or $\mathcal{O}_2$. On the system initialization, peers initialise their units (Fig. 5(a)) and retrieve static data (Fig. 5(b)). We assume that static data for data source 1 concern protected areas (e.g. national parks, fishing areas, etc.) with the corresponding geometries, while static data for data source 2 concern vessels (e.g. MMSI, vessel type, vessel name etc.). When the static data are retrieved and the first update is triggered, the ontology $\mathcal{O}_1$ will be updated with the information that vessel `v1` is located at point `p1` with latitude and longitude ϕ, λ at time `t1`. The instance matching function investigates whether there are any individual correspondences to other units (Fig. 5(c)). Indeed, given that $v1$ in data source 1 has MMSI value `xyz`, a correspondence between `v1` in $\mathcal{O}_1$ and `u1` in $\mathcal{O}_2$ is established. Next, peers request an update of positional data from the data sources (Fig. 5(d)). Spatial relations are computed and it is asserted that the position `p1` at time `t1` is within a fishing area `a1`, i.e. `(p1,a1):within` (Fig. 5(e)). Given that `v1` is of type `Tanker` and $\texttt{FishingArea} \sqsubseteq \texttt{ProtectedArea}$ and $\texttt{DangerousVessel} \equiv \texttt{Tanker} \sqcap \exists\texttt{at}(\exists\texttt{within.ProtectedArea})$, the $E-\mathcal{SHIQ}$ distributed reasoning mechanism infers that `v1` is a dangerous vessel (Fig. 5(f)).

The next examples present cases where the recognition of events require the combination of information from different units and the $E-\mathcal{SHIQ}$ distributed reasoner.

Example 2 (Fishing area violation)**.** This example, shows a more elaborated example of reasoning, where spatial relations and the combination of information in different units is necessary

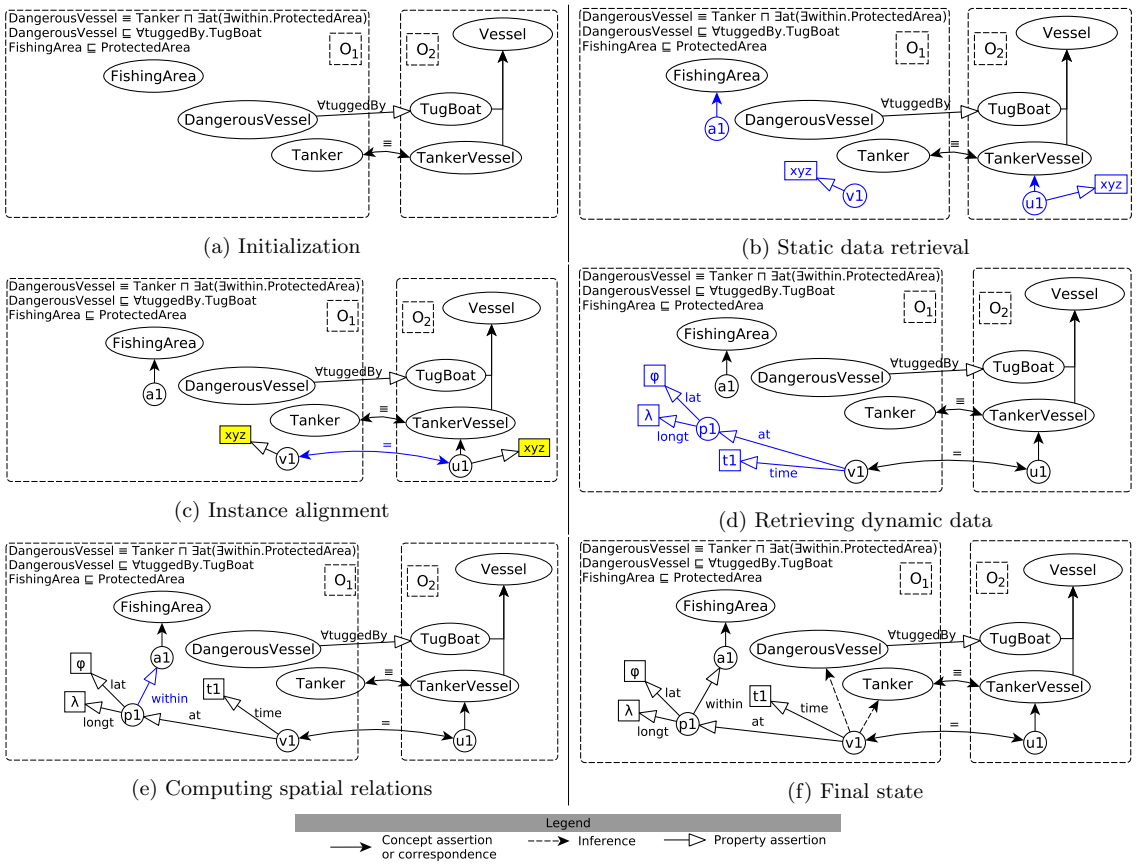

Fig. 5. Reasoning with maritime data example.

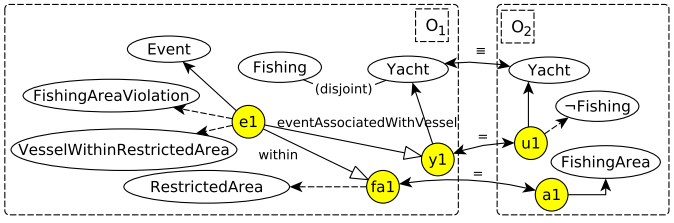

Fig. 6. Reasoning for detecting fishing area violation.

for the detection of a high level event. Fig. 6 demonstrates the combination of information from different units towards detecting high level events. We consider the high level event `FishingAreaViolation`, which is defined as:

$$\texttt{FishingAreaViolation} \equiv \texttt{Event} \sqcap \exists \texttt{EventAssociatedWithVessel}.(\neg \texttt{Fishing}) \sqcap \exists \texttt{within}.\texttt{FishingArea}$$

While the event definition is provided in unit O_1, the unit O_2 holds contextual information: vessel types and fishing areas. First, it should be noticed that the instance matching function identifies the correspondences between `y1` and `u1`, as well as `fa1` and `a1`, and that the spatiotemporal relation `(e1,fa1):within` is computed. The relation `(e1,fa1):within` also triggers the inference that `e1` is an instance of `VesselWithinRestrictedArea` concept. Assuming that `u1` is a `Yacht`, the unit O_2 infers also that `u1` is in the complement of `Fishing`, since `Yacht` and `Fishing` are disjoint concepts. The correspondences for each vessel type between units O_1 and O_2 under $E-\mathcal{SHIQ}$, enable unit O_1 to also infer that `y1` is in the complement of `Fishing` concept. In the same time, a correspondence between `RestrictedArea` and `FishingArea` allows unit O_1 to infer that `fa1` is an instance of `RestrictedArea`. Since `e1` is an event happening within a restricted area, and it is associated with an non-fishing vessel, by the definition of the `FishingAreaViolation`, `e1` it is inferred to be also an instance of `FishingAreaViolation` concept.

The next example shows the exploitation of spatiotemporal relations computed among entities.

Example 3 (Suspicious area detection)**.** In our experience with real maritime data, we have seen some vessels turn off their transmitters when moving in stealth. This behaviour can be considered suspicious, thus we have the definition:

$$\texttt{SuspiciousArea} \equiv \exists \texttt{areaAssociatedWithEvent}.\texttt{VesselInLostCommunication} \sqcap \geq 2\ \texttt{nearby}.\texttt{VesselInLostCommunication}$$

i.e. suspicious is considered any area that is associated with at least three nearby simple events of type `VesselInLostCommunication`. This example, demonstrates reasoning beyond the OWL2QL profile, since it uses a qualified cardinality restriction. As in previous examples, we assume the units O_1, O_2 illustrated in Fig. 7, where O_1 is responsible for detecting events and O_2 holds contextual information. Again, nearby and within spatiotemporal relations have been computed. In this example, we assume that `e1`, `e2`, `e3`, are events happening in spatiotemporal proximity (i.e. associated via the nearby relation) of type `VesselInLostCommunication`. If at least one of `e1`, `e2`, `e3`, is found to be within an area `x1`, e.g. `(e1,x1):within`, the reasoner will infer that also `(x1,e1):areaAssociatedWithEvent` holds. Given the

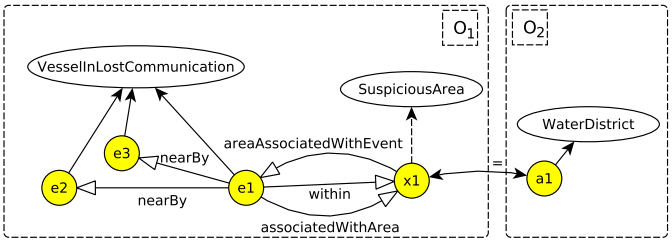

Fig. 7. Reasoning for detecting suspicious areas.

nearby relations with e2, e3, the reasoner will infer that the area x1 is a SuspiciousArea.

4.3. Balancing between OBDA and reasoning using triple templates: experimental results.

In this section we evaluate the use of triple templates towards balancing between efficient OBDA and distributed $E-\mathcal{SHIQ}$ reasoning, while populating the ontology units. Specifically, we evaluate the use of triple templates by measuring the efficiency of retrieving simple events and the efficiency of realising simple events as ontology individuals and checking the consistency of the distributed knowledge base. These tasks for simple events suffice to show that efficiency in data retrieval is achieved while also achieving efficiency in reasoning tasks.

Simple events associate the ID of a vessel with the type of the event, the time and the position where this event occurred. These events concern numerous vessels for a large time period in the Aegean Sea. As alternatives to the triple templates approach, we measure the efficiency of "Simple" and "Conditional" approaches explained below.

Simple: This approach uses a simple OBDA mapping. Given a SPARQL query for a time period, it retrieves all the simple events and populates the ontology with the raw data, inferring the type of each event, and checking the consistency of the distributed ontology. Specifically, this approach uses the following mapping:

Source:
```
SELECT id, time, lon, lat, status FROM R1
```
Target:
```
simpleEvent{id}_{time}_{lat}_{lon}_{status} :status
{status};
:associatedTime :TimeObject{time} ; :associatedWithArea
:SpObject{lat}_{lon} ; :eventAssociatedWithVessel
:vessel{id} .
```

Conditional: This approach uses an OBDA mapping for each simple event type (represented by an ontology concept), recognising the type of the event using a WHERE clause. For example, for the simple event Turn the value of the property :status should be "3" and the OBDA mapping is:

Source:
```
SELECT id, time, lon, lat, status FROM R1
WHERE status = '3'
```
Target:
```
:eventTurn{id}_{time} a :Turn ;
:associatedTime :TimeObject{id}_{time} ;
:associatedWithArea :SpObject{lat}_{lon} ;
:eventAssociatedWithVessel :vessel{id} .
```

Template: This approach uses the Simple OBDA mapping, but the realization of instances' types is done using *triple templates*. For instance, the template for simple events of type Turn, is as follows:

Table 2
Retrieval time.

T	#Events	Simple	Conditional	Template
600	470	0.811	22.891	8.522
1200	1260	0.866	36.274	32.021
1800	2106	0.950	54.067	50.615
2400	2850	0.977	141.130	69.559
3000	3507	1.026	195.985	97.022
3600	4541	1.213	245.263	120.648
4200	5481	1.272	313.324	155.622
4800	6619	1.384	388.991	193.278
5400	7198	1.542	433.557	212.441
6000	8242	1.777	509.294	252.974

Table 3
Reasoning time.

Update	#Events	Simple	Conditional	Template
600	470	43.477	1.241	0.499
1200	1260	407.476	0.902	0.860
1800	2106	507.258	1.032	1.363
2400	2850	965.218	1.860	2.174
3000	3507	3096.651	2.175	1.912
3600	4541	5298.594	2.926	3.218
4200	5481	7346.905	2.918	3.777
4800	6619	88,071.777	3.316	4.538
5400	7198	111,355.552	4.116	42.916
6000	8242	166788.949	4.313	45.668

Table 4
Overall time.

Update	#Events	Simple	Conditional	Template
600	470	44.288	24.132	9.021
1200	1260	408.342	37.176	32.881
1800	2106	508.208	55.099	51.978
2400	2850	966.195	142.990	71.733
3000	3507	3097.677	198.160	98.934
3600	4541	5299.807	248.189	123.866
4200	5481	7348.177	316.242	159.399
4800	6619	88,073.161	392.307	197.816
5400	7198	111,357.094	437.673	255.357
6000	8242	166,790.726	513.607	298.642

```
ASSERT ?e a :Turn
PREFIX : <http://www.aminess.eu/ontology/events#>
SELECT * WHERE
{
?e :associatedWithArea ?x .
?e :associatedTime ?t .
?e :status ?k .
?t :EventStartEpoch ?time .
FILTER ((?time >= ''##S##'') && (?time < ''##T##'')).
FILTER (?k = ''3'')
}
```

where the variables ##S##, ##T## represent the starting and terminating time instants of the event.

For each experiment we record the data retrieval time and the reasoning time. We evaluate the scalability of each alternative w.r.t. the amount of retrieved data, varying the data update period T in the query. Specifically, experiments concern time periods from 10 min to 100 min with a step of 10 min. Tables 2, 3, 4 present the retrieval, reasoning and overall time, respectively, for each alternative approach. The first column for each table shows the Update time period T in seconds, the second column indicates the number of events retrieved, and the third, fourth and fifth columns indicate the time required from each approach, in seconds.

The results for the retrieval time presented in Table 2 indicate that the Simple approach has a great advantage over the other two approaches, as expected. A comparison between the Simple and the Conditional approaches, shows the overhead of using WHERE

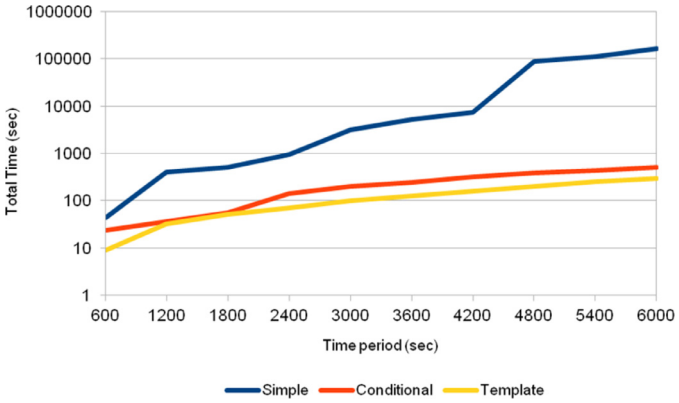

Fig. 8. Reasoning performance (retrieval+reasoning times) of three approaches.

clauses in the OBDA mapping. Similarly, a comparison of the results achieved by the Simple versus those achieved by the Template approach indicates the overhead of processing the template query. These results indicate that the Template approach is preferable over the Conditional. This can be explained by the fact that mappings for the Conditional approach are more complex compared to the other two approaches, resulting in more complex queries to the data source. On the other hand, for Simple and Template approaches, the mappings are very simple.

The time required for reasoning is shown in Table 3. Clearly, the Simple approach is not scalable, thus it should be used only for solutions where reasoning is not applied. We observe that for an update time period of 50 min, the reasoning time required is approximately 52 min (3098 s), thus for this amount of data, this approach has no practical use. On the other hand, the Conditional and Template approaches show similar performance, as expected.

Finally, the results of Table 4 provide the overall time required for data retrieval and reasoning. It is clear that for all the cases, any of the Conditional or Template approaches can be used, because they require considerably less time compared to the update period. However, the Template approach shows the best performance in every case (also shown in Fig. 8).

5. Large-scale evaluation of the OBDAIR framework

We have evaluated the implemented OBDAIR framework over a large set of maritime and weather data recorded for the Aegean Sea. Specifically, the evaluation dataset of static and dynamic maritime data in numbers is: Static data involve 452 positions and names of ports, 48 restricted regions (e.g. fishing areas, Natura 2000), as well as ID codes and characteristics of 38,530 registered vessels of various types (Yacht, Oil Tanker, Bulk Carrier, Fishing, General Cargo, Passenger Ship). Fig. 9 illustrates the ports and protected areas (in red) in the evaluation dataset. It must be pointed out that we have defined a restricted area in the center of the Aegean Sea to simulate situations of vessels entering restricted zones. The simulated area is significantly large, so that it will be related to a large number of vessels, requiring the recognition of many incidents (approximately one per vessel). This “stresses” the system to evaluate a large number of complex events’ patterns. Table 5 shows the number of vessels in each category.

The dynamic data in the evaluation dataset consist of critical points derived from AIS messages of registered vessels and weather forecasts. The critical points describe the position and status of vessels from May 1st 2015 to Sept. 1st 2015, resulting in 2,745,776 records in the database. Weather reports include water and air temperature, visibility, swell height, cloud coverage, humidity, precipitation, wind speed and direction, for different places in

Fig. 9. Ports (yellow points) and restricted areas (red polygons). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
Vessels in evaluation dataset by type.

Vessel type	Number of vessels
Yacht	620
Fishing	1770
PassengerShip	2430
BulkCarrier	7550
OilTanker	10,880
GeneralCargo	15,280

the evaluation region. We have simulated bad weather conditions using low value thresholds on visibility and swell height conditions, resulting to 62 regions that exist during the whole evaluation period.

Static data are retrieved using specific queries for each type of vessel, ports and protected areas. This allows parallel transactions for the population of the knowledge base with static data: For the evaluation tests, we have set a maximum of 5 concurrent threads[10] for retrieving static data.

Dynamic data were retrieved using query templates for weather data and critical points regarding vessels’ status. Specifically, bad weather data instances were realised by means of the following query using a triple template:

```
ASSERT ?w a :BadWeatherData
PREFIX : <http://...
/aminessLocalOntoDynamicData#>
SELECT * WHERE {
?w a :MarineWeatherData;
:weatherConditionDate ?fdate;
:weatherConditionTime ?time;
:weatherMinTemperature ?min_temperature;
:weatherMaxTemperature ?max_temperature;
:weatherConditionWindSpeedKm ?wind_speed_kmph;
:weatherConditionWindDirection ?wind_dir_degree;
:weatherConditionVisibility ?vis;
:weatherConditionSwellHeight ?swell_height_m;
:weatherCloudCover ?cloud_cover;
:weatherConditionLat ?latitude;
:weatherConditionLong ?longitude;
:hasGeometry ?geom .
FILTER((?vis < 9) || (?swell_height_m > 1)) .
FILTER(str(?fdate) = ''##DATE##'' && str(?time) =
''##WEATHERTIME##'') .}
```

[10] The number of threads is set after experimentation. Lower number of threads lead to a more serial data access, higher number of threads increase the load of concurrent requests to the RDBMS. The number of threads depends on hardware/system configuration and the RDBMS settings.

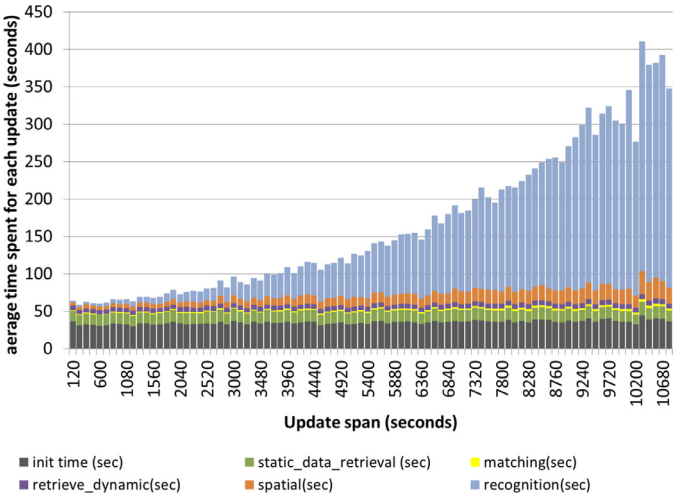

Fig. 10. Time spent for each stage of OBDAIR update cycle, per update period.

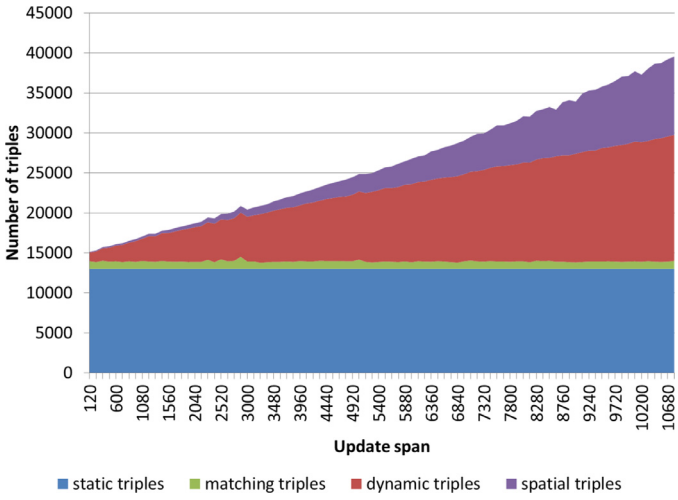

Fig. 11. Number of triples for each stage in the system's update cycle, per update period.

Dynamic data for vessels in high speed were retrieved using the following query, where a threshold of 46.3 knots was used to define high speed:

```
ASSERT ?x a :SpatialObject
ASSERT ?e a :VesselInHighSpeed
PREFIX : <http://www.aminess.eu/ontology/events#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT * WHERE {
?e :associatedWithArea ?x .
?e :movingObjectEventSpeed ?speed .
?x :hasGeometry ?geom .
?e :EventStartEpoch ?t1 .
FILTER ((?t1 ≥ ''##S##''^^xsd:long) && (?t1 <
''##T##''^^xsd:long))
FILTER (?speed > 46.3)}
```

The implemented system has been parameterised to retrieve dynamic data for a variety of update periods *T*. The OBDAIR update task includes the following stages: the retrieval of data, the computation of matching instances, the computation of spatiotemporal relations among entities and reasoning for the recognition of complex events. We have evaluated each stage of the implemented system for a variety of update periods, ranging from 120 to 10,800 s (3 h). Experiments for each update period have been repeated 20 times regarding different periods within a year, to eliminate the possibility of biased results. Thus, the evaluation results provided record the average of measurements for these experiments.

During the experiments we measure the time required for the following:

- System's initialization (init time).
- Retrieval of static data (static data retrieval).
- Retrieval of dynamic data (retrieve dynamic), this is also the time needed for extracting simple events using triple templates,
- Matching of vessels' instances (matching),
- Computation of spatial relations (spatial),
- Reasoning for the recognition of complex events,
- Updating the knowledge base, as a result of summing the above times, except the init time (update).

The results are shown in Fig. 10.

We also record the number of triples regarding:

- static data retrieved (static triples),
- dynamic data retrieved (dynamic triples),
- equalities and correspondences computed by the matching functions (matching triples),
- spatiotemporal relations (spatial triples).

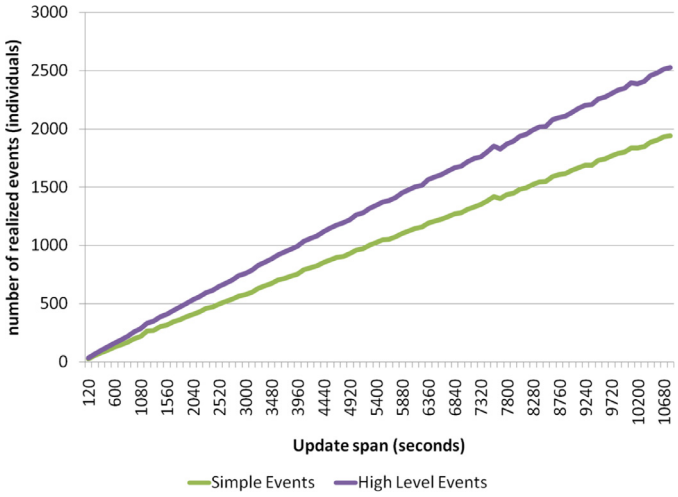

Fig. 12. Number of simple and complex events, per update period.

The recorded number of triples are shown in Fig. 11.

We finally record the number of simple and complex events per update period. The simple events are realized while retrieving dynamic data, while complex events require reasoning with the entire knowledge base. Fig. 12 illustrates the number of events (simple and complex) detected for each update period. The average number of events for each update period vary from 34.5 (26 simple and 8.5 complex events) for an update period of 120 s to 2523 events (1941 simple and 582 complex events) for an update period of 10,800 s. We observe that for each update period, the ratio of complex events to the total number of events (simple plus complex events), is approximately 23.3%. This indicates that the reasoner has a significant role in each update cycle: Approximately 23.3% of each update is completed by the reasoner.

Experiments have been conducted with a 4 peers configuration, on a Qemu Virtual Machine, with 4 64-bit processors at 2.4 GHz, 4GB memory and Java maximum memory allocation pool set to 3GB.

The experimental results in Figs. 10, and 11 show that the time required for the system initialization and the static data retrieval is approximately constant for any update period, as it is also shown in Fig. 11. This also holds for the triples computed by the instance matching functions. We observe that the time required for the retrieval of dynamic data (retrieve_dynamic in Fig. 10) re-

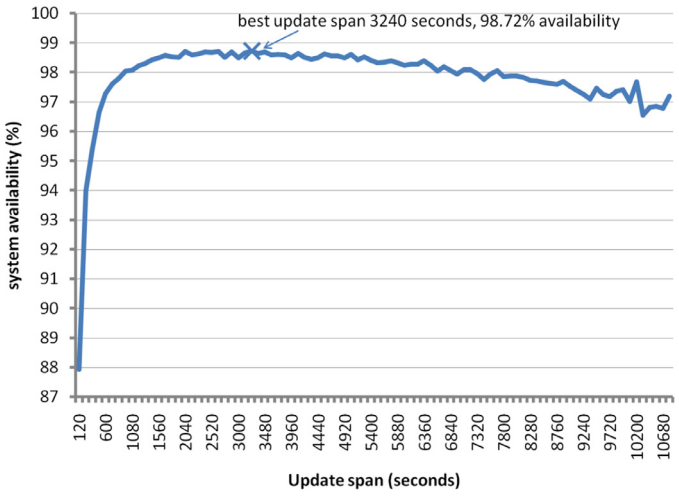

Fig. 13. System availability as percentage of the update period.

mains small compared to the overall time required for the update. This observation indicates the scalability of Ontop OBDA: Although, as shown in Fig. 11 dynamic data triples increase as the update period increases, dynamic data retrieval time has lower increase rate than the increase rate of the amount of data retrieved. Also, Fig. 11 shows that the spatio-temporal relations computed (spatial triples) are proportional to the dynamic triples retrieved, as expected. However, Fig. 10 indicates that the distributed computation of these triples scales well since, as the update period increases, the time required for these computations has low increase rate. This is due to the distribution of these computations to distinct workers. Finally, we observe that for update periods above 3960 s the time required for the recognition of complex events (recognition time) is more than the time needed for all the other tasks in the update process. This indicates that the complex events recognition stage is the most demanding phase for each update.

If we consider that the implemented OBDAIR instance updates its knowledge base on constant time intervals, the experimental results can aid the decision for the optimum update period. Specifically, the update time period s has to be always larger than the time $u = f(s)$ required for the system to perform a complete update for the data in s. The difference $s - u$ shows the time that the system is available to do other tasks (e.g. to serve queries on the updated data, or to perform further reasoning on the data). The maximum availability percentage $max_s\{\frac{s-u}{s} * 100\%\}$ indicates the best update period for a specific set of data sources and OBDAIR configuration: This is the update period which allows the greatest system availability. Fig. 13 indicates that in our case, the optimum update period is 3240 s (54 min), with 98.72% percent system availability (i.e. only 1.28% of the update time period is required for the system update). We also observe, that the minimum availability recorded is 87.93% in our experiments. This indicates that in case more efficient distributed or parallel reasoners are used, then, given that the reasoning task is the most demanding of all tasks, the OBDAIR would scale more effectively, increasing further the availability time.

Concluding this section, the implications of experimental results are as follows:

- Distribution of knowledge, of data retrieval, data integration and data processing tasks can result to very efficient solutions, even in large update periods where more (static and dynamic) data are retrieved and processed, resulting to a large number of spatiotemporal relations (shown in Figs. 10–12). This implies that OBDAIR provides a framework for efficient retrieval and integration of data from different RDBs. It must be noticed that even in small update periods (e.g. for frequently updated data sources) OBDAIR is very efficient, with high availability.
- Distribution of reasoning provides efficient solutions to reasoning with data, especially when this is the more demanding task in each update cycle: This is shown by the availability of the system as percentage of the update period duration (Fig. 13). High availability means that the time the system requires to reason with the data is much less than the update period duration. Indeed, the system scales well, since it has the potential to increase its availability drastically as the update period duration increases from 120 to 1200 s, while, the availability the system is not reduced considerably as the update period duration increases from 2000 to 10,700 s. This is an important result, given the increase of the number of triples and events per update period shown in Figs. 11, 12.

6. Related work

Several related works have been investigating the research topics of this paper, individually or partially combined, i.e. (a) large-scale and near real-time recognition of complex events related to moving entities, (b) ontology-based access, integration and exploitation of disparate and heterogeneous (dynamic/static) data, (c) distributed reasoning in large-scale modularized and expressive knowledge bases. To the best of our knowledge, OBDAIR framework is the only work that integrates existing frameworks, technologies and new methods towards addressing all three topics conjunctively, aiming at efficient semantics-based complex event recognition. In this paper, we place our framework in the intersection of those works, presenting the most representative and related ones from each research topic.

A recent review paper on semantic complex event processing (Marc et al., 2012), concludes that a number of approaches to integrating Semantic Web technologies into complex event processing (CEP) have been proposed. These approaches can take advantage of the reusability and extensibility of ontologies, and at the same time, the aggregation and analysis of event information that is generated by different applications can be less difficult and time consuming. Furthermore, according to authors, semantics-based CEP engines dealing with ontologies can achieve better and faster results in event matching and can collect more complex events with more semantics. However, there is still enough work to do, especially in the area of optimization of event matching strategies.

In Rinne, Blomqvist, Keskisärkkä, and Nuutila (2013), authors propose an event processing ontology design pattern for representing and reasoning over complex and composite event objects, progressing the research area of RDF stream processing. They demonstrate generic query patterns for event object management using SPARQL 1.1, which facilitates event processing. The proposed model is aligned to existing standards, such as the SSN ontology, and it is compatible with other event models, such as Event-F (Scherp, Franz, Saathoff, & Staab, 2009). There are no any reference however on disparate and heterogeneous data integration, on big-data issues, or on advanced and parallel reasoning capabilities.

Early ontology-based data integration approaches such as Zamboulis et al. (2008), Vandecasteele and Napoli (2012a) address the integration problem using a single monolithic ontology describing the available data, also without exploiting the advantages of an OBDA framework. Furthermore, only recently the problem of integrating data using OBDA solutions in settings with dynamic and voluminous data has been given special attention. This problem requires optimization techniques and distributed solutions at the architectural and modelling level, in order to be able to efficiently answer translated (sparql-to-sql) ontology-mediated queries.

Optique (Haase et al., 2013) addresses the challenge of OBDA-based data integration giving emphasis to the optimization of distributed querying in order to allow scalable end-user access to big data (Giese et al., 2013).The federated query processing engine (FedX) operating on top of autonomous semantic databases enables the virtual integration of heterogeneous sources implementing efficient query evaluation strategies. Optique uses Ontop OBDA system (Bagosi et al., 2014) for mapping RDB data to a single ontology. Although plans for exploiting ontology modularization techniques have been announced (Haase et al., 2013), to the best of our knowledge there is not a specific approach towards this yet.

SPEED (Figueiredo et al., 2013) is an OBDA instantiation for a Peer Data Management System (PDMS) where queries submitted to a peer are answered with data residing at that peer and with data acquired from neighbor peers through mappings. This work put together OBDA and a visual query system to access geospatial data in a PDMS. The reformulation of queries between neighbor peers by taking into account semantic correspondences is stated as a future work. SPEED applies a non-mediated schema approach to OBDA-based data integration. In absence of evaluation results we cannot provide any further details about the scalability or the reasoning capabilities of this approach.

In Mena et al. (2000) authors present the OBSERVER Ontology Based System, focusing on a query translation approach using multiple ontologies. Queries are formulated using terms in a user-selected ontology and are translated into terms of target ontologies describing relevant information. Multiple ontologies access heterogeneous, distributed and independently developed data repositories translating queries expressed by means of description logics expressions to the local query languages of data repositories. According to authors, although scalability and reasoning issues have been considered in a great extent, the access to the distributed data repositories is a bottleneck to the performance of the prototype system.

Aiming to provide access to data using a common data model and a common query language, the Khaos Ontology-based Mediation Framework (KOMF) for the development of ontology-based data integration mediators (an en extension of SD-Data) (Navas-Delgado & Aldana-Montes, 2009) provides a semantically coherent framework for representing and accessing data from various sources through queries to the mediating view. This work is not related to OBDA, nor it handles issues such as processing of data near to the sources, and distributed data access and reasoning.

Although, compared to OBDAIR, the same differences appear in the ontology-based data integration approach for web analytics in e-commerce that is proposed in García, García-Nieto, and Aldana-Montes (2016), the aim there is to collect, integrate and store web analytics data, from many sources of popular and commercial digital footprints. Such data is enriched and semantically annotated in order to train data mining procedures for analysis of customer behavior in real e-commerce sites. The approach is using a single ontology as a mediated schema, with custom data to ontology mappings via build-in wrappers. Interestingly, this work has been evaluated for the integration of data beyond RDBs i.e., integrating data provided also in JSON and JavaScript (.js) format.

Karma system (Harth, Knoblock, Stadtmüller, Studer, & Szekely, 2013) aims at automating the process of mapping data sources to an ontology, emphasizing the modeling of both static and dynamic sources and the integration of new data sources. In close relation to our aims, Karma is able to automatically transform data to RDF using the R2RML model, providing access to the available sources via the Data-Fu Program (Stadtmüller, Speiser, Harth, & Studer, 2013). This collects data and evaluates queries over the data, while taking into account the semantics of ontology constructs. In contrast to our approach, a unique shared domain ontology is used to model the data sources and to generate the RDF-linked graphs. Their evaluation efforts report a scalable Data-Fu interpreter that scales from small sources (hundreds of triples) to sources of moderate size (hundreds of thousands of triples). However, there are no specific reports on how they handle advanced/parallel reasoning capabilities nor on the introduction of advanced methods for processing data close to the data sources (in-situ processing).

To the best of our knowledge, related work on ontology-based maritime event recognition (e.g. Vandecasteele & Napoli, 2012a; 2012b) approach the problem in a centralized manner, using a monolithic ontology for modelling maritime and sensor data. This is in contrast to the aim of our approach to meet the important requirement of allowing different data source providers to model their own data in a flexible and autonomous manner without having prior knowledge of others' models, supporting the (semi-)automated integration of these data and models. Furthermore, these approaches require a data fusion pre-step, and moreover, they require the continuous ingestion (update) of data into a single OWL model, in contrast to our multiple and coupled ontologies approach. Moreover, these systems use an extension of the SWRL rule language for detecting events but the creation of these rules (as stated by their authors) can further constrain domain experts responsible for creating the model. Finally, there are no any evidences that these approaches consider scalability issues or incorporate frameworks for advanced/parallel reasoning capabilities.

Beyond the maritime domain, a wide range of ontology-based data access applications have been developed, from logistics and transportation of goods e.g. Casu, Cicala, and Tacchella (2013) to biology and biomedical e.g. Hoehndorf, Slater, Schofield, and Gkoutos (2015) and Livingston, Bada, Baumgartner, and Hunter (2015) domains. Yet, these applications follow the common approach that clients have access to the data from a single ontology, which eventually affects scalability on reasoning tasks and constraints data source providers on the way they model their own data. Although, scalability and time-efficiency measurements are not available for comparison, the need to semantically integrate disparate and heterogeneous data sources, in conjunction to recent advancements in distributed knowledge bases and reasoning (Santipantakis & Vouros, 2014), prove that the use of a single ontology model and reasoning capabilities in large-scale (big data) systems do not offer the advantages of distributed knowledge bases.

7. Discussion

OBDAIR provides a framework towards scalable data-driven domain-specific applications that support decision-making and problem-solving by exploiting data from disparate and heterogeneous RDBs. This is achieved via its ability to access, process and reason with data in a distributed/decentralized but still unified manner. Beyond its advantage of avoiding any replication of data in additional stores, the framework emphasizes the introduction of close to the data sources (in-situ) processing of data to effectively support data integration and data extraction for the dynamic population of the distributed knowledge base. While data integration tasks of OBDAIR are necessary for associating entities and coupling information from different data sources, information extraction aims to identify information that is necessary for the efficient performance of reasoning tasks. Thus, as is has been pointed out, information extraction and integration tasks precede any reasoning task, so as to support efficiency, preserve coherence of specifications in the distributed knowledge base, and thus completeness and soundness of the reasoning tasks. Thus, the functionality of this layer is highly dependent on the modular ontology framework used, on the semantics of the inter-unit associations and on the ontology specifications involved in any reasoning task.

Table 6
An estimate of human effort for system update tasks. Dependencies among tasks show what is required to be done for the completion of each OBDAIR setup task.

ID	Task	Work load/demand in % of overall time	Dependencies with tasks
1	Develop ontology units (one for each corresponding RDB source)	10% - Low demanding, given knowledge engineers' expertise. This effort may be further minimized by means of ontology learning methods from data	3, 4, 5
2	Develop ontology unit for the high-level (complex event patterns)	25% - Highly demanding due to the need for knowledge elicitation - collaboration with domain experts to specify patterns of complex events	7
3	Specify Ontop mappings (RDB data to ontology unit)	30% - This is the most demanding task: It requires learning the mapping language and to incorporate functions needed for in-situ processing/transformation of data types/formats	1
4	Specify COWL correspondences among ontology units	5% - The least demanding task, which may be further minimized by using ontology mapping methods	12
5	Specify triple templates	10% - Less demanding task, based on previous knowledge of SPARQL	1,2,3
6	Connection to new RDB	0% this task demands the sum of dependent tasks' demands	1,3,4

To configure and setup OBDAIR towards data driven approaches for efficient reasoning and decision-making, there are a number of tasks that require human effort: Development of domain and data source - specific ontology units, specification of mappings between data and ontology units and between ontology units themselves, specification of triple templates for the population of the distributed knowledge base, specification of patterns for data integration (linking entities), and implementation of functions for computing relations among entities. This issue is discussed among the limitations of OBDAIR. Nevertheless, it must be pointed out that in OBDAIR, as in related approaches, there is space for minimizing human-involvement by automating tasks such as, learning ontologies from data, computing mappings between data and ontology units, computing links between entities.

According to the above, the strengths of our proposal for OBDAIR are as follows:

- Most of the current paradigms for accessing data sources by means of ontologies use a single ontology. The practice of using a single, monolithic ontology to represent, manage and reason with data from different data sources, presents limitations to efficiently exploit large and disparate data sources. OBDAIR uses a distributed knowledge base (modular ontology) for exploiting data from disparate and heterogeneous data sources.
- In contrast to related approaches, OBDAIR builds on methods for processing data near to the sources, so as to populate ontologies with information that is necessary for reasoning (in contrast to raw data), saving reasoning time.
- The above, in combination to addressing data retrieval, integration, processing and reasoning in an inherently distributed setting results to efficiency.

The limitations of OBDAIR are as follows:

- OBDAIR requires human effort to build a coherent solution for accessing, integrating and reasoning with data. Specifically, the human effort required for the different tasks needed to configure OBDAIR to a new domain, and/or to changes in the data sources exploited, is described and quantified in relation to the total time needed to set up the framework for a specific application, in Table 6.
  Specifically, the integration of a new RDB data source requires adding effort for tasks (1), (3), and (4). This effort may be reduced if there are similarities to other sources. For instance, in order to integrate a new RDB source storing data related to vessels' information, existing ontology specifications (task 1), existing specifications of RDB to RDF mappings (task 3), and existing COWL correspondences (task 4), can be reused and refined as needed.
- The current implementation of OBDAIR integrates data that reside only on RDBs, so any other format (e.g., .csv, .xml) must be either imported to an RDB or the incorporation and/or development of RDF generators is needed.
- Efficient methods for link discovery and data integration need to be incorporated in the overall process to enhance the efficiency of these tasks w.r.t. proved properties for the precision/recall achieved.
- As experimental results show, there is space for improvement in case more efficient reasoners are used: This is also necessary if demanding reasoning tasks (e.g. spatiotemporal reasoning) are required.

The evaluation of the OBDAIR instance for recognising events in the maritime domain has revealed that using a distributed knowledge base (modular ontology) approach for accessing data by means of Ontop and reasoning with the data by means of $E-\mathcal{SHIQ}$ can be highly efficient for data sources of large volume, with dynamic updates and with heterogeneous data formats. These are important results towards generating the necessary knowledge for the detection of high-level and critical complex events in near-to-real-time. Thus, albeit the limitations discussed above, this work shows the potential of OBDAIR to address data retrieval, integration and reasoning from heterogeneous and dynamically updated data sources, jointly and in computational efficient ways, while it provides opportunities for incorporating advances in various topics of research.

8. Conclusions and future work

This paper presented the OBDAIR distributed system for accessing, integrating and reasoning with data from disparate and heterogenous data sources. Specifically, OBDAIR

- Supports the distribution of knowledge to peers which are responsible for accessing, integrating and reasoning with data from disparate and heterogeneous data sources,
- Incorporates methods for processing data "close to the data sources", i.e. on retrieval, so as to populate ontologies with extracted information (in contrast to raw data), saving reasoning time,
- Provides support for efficient data retrieval, information integration and reasoning tasks, by means of inherent support to parallelization.

The presented and evaluated instantiation of OBDAIR integrates the Ontop OBDA framework and the distributed knowledge representation framework $E-\mathcal{SHIQ}$.

The proposed framework has been instantiated for detecting complex events in the maritime domain and has been evaluated by means of extended experiments on real-world datasets.

As discussed in Section 6, to a greater extent than other approaches, OBDAIR addresses retrieval and integration of data from disparate and heterogenous data sources, together with near-to-the-sources processing of data and reasoning with the data, *jointly*: This happens in an efficient way due to the distribution of knowledge, and to the parallelisation of tasks that is inherently enabled in the framework. The major implication of the proposed approach is that processing of data near-to-the-sources following the in-situ paradigm of data processing, together with distribution of knowledge and tasks, can result to highly efficient analysis solutions, even for dynamically updated data sources. This is an important implication towards implementing data-driven intelligent systems for supporting decision making is various domains.

Experimental results show the computational efficiency that can be achieved for detecting complex events in large geographic areas in the maritime domain. As experimental results reveal, the reasoning task for the recognition of complex events is the most demanding task during each update cycle: However this does not affect the overall system efficiency, and we may address this issue by finding the appropriate update time period w.r.t. the requirements of the reasoning task and the update frequency of data sources.

Plans for future work include the evaluation of the proposed framework for larger networks of peers, where peers are not only associated with a specific ontology unit but for a domain-specific segment of the data available (e.g. for specific geographical areas). Issues for balancing the workload between peers is important towards incorporating streaming data sources as well. Furthermore, we plan to investigate methods that will enable the detection of individual correspondences using more sophisticated instance matching methods (e.g. similar to those reported in Yao, Gao, and Wang (2003) or Ngomo and Auer (2011)) exploiting also the distribution of tasks to peers. In close relation to that, we also aim to interlink individuals by means of detecting domain specific relations between instances within a unit as well as between units. Regarding the spatiotemporal aspects, which are of particular interest to several important domains we are also in the process of studying the potential of extending the peers with spatiotemporal reasoning capabilities so as to take advantage of the inherent parallelism. Finally, we plan to evaluate the implemented system for the recognition of more complex events, investigating the use of more expressive events' patterns.

Acknowledgements

This work was carried out in the context of the project AMINESS: "Analysis of maritime information for Environmentally Safe vessel shipping" funded by the General Secretariat for Research and Technology (GSRT) grant agreement n° BENM9-ZMH / 21/05/2013, Greece (http://aminess.eu/). We would like to thank the reviewers and the editor for their valuable comments and suggestions towards improving our paper.

References

AIS (2016). Automatic Identification Systems (AIS). http://www.imo.org/OurWork/Safety/Navigation/Pages/AIS.aspx. Accessed: 2016-05-03.

Arenas, M., Bertails, A., Prud'hommeaux, E., & Sequeda, J. (2012). A direct mapping of relational data to rdf. https://www.w3.org/TR/rdb-direct-mapping/. Accessed: 2016-05-0.

Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., … Xiao, G. (2014). The Ontop framework for ontology based data access. In *The semantic web and web science* (pp. 67–77). Springer.

Bao, J., Voutsadakis, G., Slutzki, G., & Honavar, V. (2009). Package-based description logics. In *Modular ontologies* (pp. 349–371). Springer.

Bizer, C., & Seaborne, A. (2004). D2RQ-treating non-RDF databases as virtual RDF graphs. In *Proceedings of the 3rd international semantic web conference (ISWC2004)* http://www.wiwiss.fu-berlin.de/suhl/bizer/pub/Bizer-D2RQ-ISWC2004.pdf..

Bouquet, P., Giunchiglia, F., Van Harmelen, F., Serafini, L., & Stuckenschmidt, H. (2003). C-OWL: Contextualizing ontologies. In *The semantic web-ISWC 2003* (pp. 164–179). Springer.

Brüggemann, S., Bereta, K., Xiao, G., & Koubarakis, M. (2016). Ontology-based data access for maritime security. In H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S. P. Ponzetto, & C. Lange (Eds.), *The semantic web. Latest advances and new domains: 13th international conference, ESWC 2016, Heraklion, Crete, Greece, May 29 – June 2, 2016, proceedings* (pp. 741–757)). Cham: Springer International Publishing. doi:10.1007/978-3-319-34129-3_45.

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., … Xiao, G. (2017). Ontop: Answering SPARQL queries over relational databases. *Semantic Web Journal, 8*(3), 471–487. doi:10.3233/SW-160217.

Casu, M., Cicala, G., & Tacchella, A. (2013). Ontology-based data access: An application to intermodal logistics. *Information Systems Frontiers, 15*(5), 849–871.

Chen, L., & Nugent, C. (2009). Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems, 5*(4), 410–430.

Civili, C., Console, M., Giacomo, G. D., Lembo, D., Lenzerini, M., Lepore, L., … Savo, D. F. (2013). MASTRO STUDIO: Managing ontology-based data access applications. *PVLDB, 6*(12), 1314–1317.

Das, S., Sundara, S., & Cyganiak, R. (2012). R2RML: RDB to RDF mapping language. http://www.w3.org/TR/r2rml/. Accessed: 2016-05-03.

Eisenberg, V., & Kanza, Y. (2012). D2RQ/update: updating relational data via virtual RDF. In *Proceedings of the 21st international conference companion on World Wide Web* (pp. 497–498). ACM.

Figueiredo, R., Pitta, D., Salgado, A. C., & Souza, D. (2013). Geographic data access in an ontology-based peer data management system. *Journal of Information and Data Management, 4*(2), 146.

García, M. d. M. R., García-Nieto, J., & Aldana-Montes, J. F. (2016). An ontology-based data integration approach for web analytics in e-commerce. *Expert Systems with Applications, 63*(C), 20–34. doi:10.1016/j.eswa.2016.06.034.

Ghidini, C., Serafini, L., & Tessaris, S. (2007). On relating heterogeneous elements from different ontologies. In *International and interdisciplinary conference on modeling and using context* (pp. 234–247). Springer.

Giese, M., Calvanese, D., Haase, P., Horrocks, I., Ioannidis, Y., Kllapi, H., … Waaler, A. (2013). Scalable end-user access to big data. In R. Akerkar (Ed.), *Big data computing*. CRC Press.

Goh, C. H. (1996). *Representing and reasoning about semantic conflicts in heterogeneous information systems*. Sloan School of Management, MIT Ph.D. thesis..

Grau, B. C., Parsia, B., & Sirin, E. (2004). Tableau algorithms for E-Connections of description logics. *Technical Report*. UMIACS, Maryland http://www.mindswap.org/2004/multipleOnt/papers/EconnTableau.ps..

Haase, P., Horrocks, I., Hovland, D., Hubauer, T., Jimenez-Ruiz, E., Kharlamov, E., … Santarelli, V., et al. (2013). Optique System: towards ontology and mapping management in OBDA solutions. In *WoDOOM* (pp. 21–32).

Harth, A., Knoblock, C. A., Stadtmüller, S., Studer, R., & Szekely, P. (2013). On-the-fly integration of static and dynamic linked data. In *Proceedings of the fourth international conference on consuming linked data - volume 1034*. In *COLD'13* (pp. 1–12). CEUR-WS.org.

Hoehndorf, R., Slater, L., Schofield, P. N., & Gkoutos, G. V. (2015). Aber-owl: A framework for ontology-based data access in biology. *BMC Bioinformatics, 16*(1), 26. doi:10.1186/s12859-015-0456-9.

Homola, M., & Serafini, L. (2010). Augmenting subsumption propagation in distributed description logics. *Applied Artificial Intelligence, 24*(1–2), 39–76.

Isele, R., & Bizer, C. (2013). Active learning of expressive linkage rules using genetic programming. *Web Semantics: Science, Services and Agents on the World Wide Web, 23*, 2–15.

Kontchakov, R., Rezk, M., Rodríguez-Muro, M., Xiao, G., & Zakharyaschev, M. (2014). Answering sparql queries over databases under owl 2 ql entailment regime. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, & C. Goble (Eds.), *The semantic web – ISWC 2014: 13th international semantic web conference, Riva del Garda, Italy, October 19–23, 2014. Proceedings, Part I* (pp. 552–567)). Cham: Springer International Publishing. doi:10.1007/978-3-319-11964-9_35.

Kutz, O., Lutz, C., Wolter, F., & Zakharyaschev, M. (2003). E-connections of description logics. *International workshop on description logics (DL2003). CEUR-WS.ORG*: 81.

Livingston, K. M., Bada, M., Baumgartner, W. A., & Hunter, L. E. (2015). Kabob: Ontology-based semantic integration of biomedical databases. *BMC Bioinformatics, 16*(1), 126. doi:10.1186/s12859-015-0559-3.

Marc, S., Stella, G. G., Dennie, A., Irina, A., Arne, K., & Arne, D. (2012). Semantic complex event processing. In *Proceedings of the 5th WSEAS congress on applied computing conference, and proceedings of the 1st international conference on biologically inspired computation* (pp. 38–43).

MarineTraffic (2016). Marine traffic. http://www.marinetraffic.com/. Accessed: 2016-05-03.

Mena, E., Illarramendi, A., Kashyap, V., & Sheth, A. P. (2000). OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and parallel Databases, 8*(2), 223–271.

Motik, B., Grau, B. C., Horrocks, I., Z. Wu, A. F., & Lutz, C. (2015). OWL 2 web ontology language profiles (2nd ed.). https://www.w3.org/TR/owl2-profiles/. Accessed: 2015-06-04.

Navas-Delgado, I., & Aldana-Montes, J. F. (2009). Extending sd-core for ontology-based data integration. *Journal of Universal Computer Science, 15*(17), 3201–3230.

Ngomo, A.-C. N. (2013). Orchid–reduction-ratio-optimal computation of geo-spatial distances for link discovery. In *International semantic web conference* (pp. 395–410). Springer.

Ngomo, A.-C. N., & Auer, S. (2011). LIMES: A time-efficient approach for large-scale link discovery on the web of data. In *22nd international joint conference on artificial intelligence*. In *IJCAI'11* (pp. 2312–2317). AAAI Press.

Patroumpas, K., Artikis, A., Katzouris, N., Vodas, M., Theodoridis, Y., & Pelekis, N. (2015). Event recognition for maritime surveillance. In *EDBT* (pp. 629–640).

Pellegrini, T., Auer, S., Tochtermann, K., & Schaffert, S. (2009). *Networked knowledge-networked media: Integrating knowledge management, new media technologies and semantic systems*: 221. Springer.

Pinkel, C., Binnig, C., Kharlamov, E., & Haase, P. (2013). Pay-as-you-go matching of relational schemata to OWL ontologies with IncMap. In *Proceedings of the 2013th international conference on posters & demonstrations track-volume 1035* (pp. 225–228). CEUR-WS. org.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Rosati, R. (2008). Linking data to ontologies. In *Journal on data semantics X* (pp. 133–173). Springer.

Rinne, M., Blomqvist, E., Keskisärkkä, R., & Nuutila, E. (2013). Event processing in rdf. In *Proceedings of the 4th international conference on ontology and semantic web patterns - volume 1188*. In *WOP'13* (pp. 52–64). Aachen, Germany, Germany: CEUR-WS.org. http://dl.acm.org/citation.cfm?id=2874566.2874571..

Rodrıguez-Muro, M., Kontchakov, R., & Zakharyaschev, M. (2013). Query rewriting and optimisation with database dependencies in Ontop. *Proceedings of DL, 2013*.

Santipantakis, G., Kotis, K., & Vouros, G. (2015a). Accessing and reasoning with data from disparate data sources using modular ontologies and obda. *11th international conference on semantic systems, semantics 2015, Vienna*. ACM New York, NY, USA.

Santipantakis, G., Kotis, K., & Vouros, G. A. (2015b). Ontology-based data integration for event recognition in the maritime domain. In *Proceedings of the 5th international conference on web intelligence, mining and semantics* (p. 6). ACM.

Santipantakis, G., & Vouros, G. A. (2014). Distributed reasoning with coupled ontologies: The $E-\mathcal{SHIQ}$ representation framework. *KAIS, 41*(3), 1–44.

Scherp, A., Franz, T., Saathoff, C., & Staab, S. (2009). F–a model of events based on the foundational ontology Dolce+DnS ultralight. In *Proceedings of the fifth international conference on knowledge capture*. In *K-CAP '09* (pp. 137–144). New York, NY, USA: ACM. doi:10.1145/1597735.1597760.

Sequeda, J. F., Arenas, M., & Miranker, D. P. (2014). OBDA: Query rewriting or materialization? In practice, both!. In *The semantic web–ISWC 2014* (pp. 535–551). Springer.

Serafini, L., Borgida, A., & Tamilin, A. (2005). Aspects of distributed and modular ontology reasoning. In *IJCAI: 5* (pp. 570–575).

Serafini, L., & Tamilin, A. (2004). Local tableaux for reasoning in distributed description logics. In *Proceedings of the international workshop on description logics, dl: 4* (pp. 100–109).

Serafini, L., & Tamilin, A. (2005a). Distributed instance retrieval in heterogeneous ontologies. *SWAP*. Citeseer.

Serafini, L., & Tamilin, A. (2005b). Drago: Distributed reasoning architecture for the semantic web. In *The semantic web: Research and applications* (pp. 361–376)). Springer.

Stadtmüller, S., Speiser, S., Harth, A., & Studer, R. (2013). Data-fu: A language and an interpreter for interaction with read/write linked data. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 1225–1236). ACM.

Szekely, P., Knoblock, C. A., Yang, F., Fink, E. E., Gupta, S., Allen, R., & Goodlander, G. (2014). Publishing the data of the Smithsonian American art museum to the linked data cloud. *International Journal of Humanities and Arts Computing, 8*(supplement), 152–166.

Vandecasteele, A., & Napoli, A. (2012a). An enhanced spatial reasoning ontology for maritime anomaly detection. In *System of systems engineering (SoSE), 2012 7th international conference on* (pp. 1–6). doi:10.1109/SYSoSE.2012.6384120.

Vandecasteele, A., & Napoli, A. (2012b). Spatial ontologies for detecting abnormal maritime behaviour. In *OCEANS, 2012-Yeosu* (pp. 1–7). IEEE.

Vouros, G. (2014). Decentralized semantic coordination of interconnected entities via belief propagation. *AI Communications, On Line*. doi:10.3233/AIC-140624.

Yao, Z., Gao, L., & Wang, X. S. (2003). Using triangle inequality to efficiently process continuous queries on high-dimensional streaming time series. In *Scientific and statistical database management, 2003. 15th international conference on* (pp. 233–236). IEEE.

Zamboulis, L., Poulovassilis, A., & Roussos, G. (2008). Flexible data integration and ontology-based data access to medical records. In *Bioinformatics and bioengineering, 2008. BIBE 2008. 8th IEEE international conference on* (pp. 1–6). IEEE.

Zimmermann, A. (2007). Integrated distributed description logics. In *Proceedings 20th international workshop on description logic (DL)* (pp. 507–514). Bolzano University Press.

Zimmermann, A. (2013). Logical formalisms for agreement technologies. In *Agreement technologies* (pp. 69–82). Springer.