Preliminary extended version of a paper that appeared at AI 2015

Stream-Query Compilation with Ontologies*

Özgür L. Özçep, Ralf Möller, and Christian Neuenstadt

Institute of Information Systems (IFIS)
University of Lübeck
Lübeck, Germany
{oezcep,moeller,neuenstadt}@ifis.uni-luebeck.de

Abstract. Rational agents perceiving data from a dynamic environment and acting in it have to be equipped with capabilities such as decision making, planning etc. We assume that these capabilities are based on *query answering* with respect to (high-level) *streams of symbolic descriptions*, which are grounded in (low-level) data streams. Queries need to be answered w.r.t. an ontology. The central idea is to compile ontology-based stream queries (continuous or historical) to relational data processing technology, for which efficient implementations are available. We motivate our query language STARQL (Streaming and Temporal ontology Access with a Reasoning-Based Query Language) with a sensor data processing scenario, and compare the approach realized in the STARQL framework with related approaches regarding expressivity.

Keywords: stream, rewriting, ontology, description logic

1 Introduction

Many of the main problems, topics, and activities of AI research are related to the design and analysis of rational agents. Rational agents perceive sensor data from an environment and act therein so "as to achieve one's goals, given one's beliefs" [13, p. 7]. This hints to the challenging problem of transforming sensor data into high-level conceptualizations—required for representing and reasoning with beliefs and goals—and transforming operations on the higher level to operations on the lower level. We assume that the agent has to transform lower-level data streams to higher-level streams of symbolic descriptions.

The challenging transformation problems mentioned above are not new and have been approached in different projects. One prominent example is the DyKnow project [10], in which a stream-based middleware for knowledge processing was implemented and tested within a UAV scenario. To the best of our knowledge the approach followed in DyKnow requires making symbolic descriptions explicitly available (in a materialized form) in order to apply powerful inference and planning services needed to achieve the goals of the agents. Note that

* This work has been supported by the European Commission as part of the FP7 project Optique.

incurred scalability problems are not unique to DyKnow but apply to many logic-based situation awareness techniques [4,5]. In contrast to these, the approach of this paper tackles the problem of performant transformations by generating only "views" of high-level streams where the views are defined by declarative mappings from low-level data streams to high-level streams of symbolic descriptions. Hence, the high-level stream is actually not materialized by the mappings but is considered as a virtual stream on the abstraction level needed to accomplish (goal-based) reasoning. In order to realize inference services, such as ontology-based query answering on high-level streams, one has to reduce them to operations on low-level streams. This kind of reduction is an essential element of the paradigm of *ontology-based data access* (OBDA) [8] and involves *rewriting a query to SQL (or stream SQL such as CQL [1]) w.r.t. the ontology* as well as *unfolding a query with respect to mapping rules.*

Recent work on temporalizing and streamifying OBDA [2,6,9,7,12,11] provides solutions for adapting methods, concepts, and ideas from temporal logics and relational stream languages to the ontological layer. Nonetheless, the picture of relations between different approaches, in particular w.r.t. the semantics of specific logical operators such as the window operator, is still incomplete. To fill this gap, we give an analysis of the query framework STARQL (Streaming and Temporal ontology Access with a Reasoning-Based Query Language) in comparison to other streaming and temporal OBDA approaches.

The contributions of this paper are the following: STARQL's semantics is tailored towards a plug-in architecture. In order to extend the expressivity picture, we show that an OBDA instantiation of STARQL has a different semantics that leads to the same set of answers as w.r.t. the original semantics (Thm. 1). As a corollary we get first-order logic (FOL) rewritability for answering STARQL queries (Thm. 2). We show that (a safe fragment of) TCQs [6] is embeddable into STARQL (Prop. 3).[1]

2 Logical Preliminaries

An ontology is a triple $\mathcal{O} = \langle Sig, \mathcal{A}, \mathcal{T} \rangle$ with a signature Sig, an ABox $\mathcal{A}$ (set of assertional axioms), and a TBox $\mathcal{T}$ (set of terminological axioms). The STARQL framework allows for plugging in different DLs, but here we focus on the DL-Lite family [8], which is tailored towards the OBDA paradigm. In OBDA, query answering over an ontology is reduced to model checking on the data by rewriting the query w.r.t. the TBox and unfolding it w.r.t. mappings. In the whole paper let $[n] = \{0, 1, \ldots, n\}$ be the natural numbers up to $n \in \mathbb{N}$. A *conjunctive query* $\exists y_1, \ldots, \exists y_m . \rho(y_1, \ldots, y_m, x_1, \ldots, x_n)$ consists of a prefix of existential quantifiers and a conjunction ρ of atoms of the form $A(z)$ or $R(z, w)$ with $z, w \in \{y_1, \ldots, y_m, x_1, \ldots, x_n\}$ or z, w are constants. The y_i are bound by the existential quantifiers, the free variables x_i are the output slots of the query. If the set of free variables is empty, the query is called Boolean. We use letters α, β for CQs

[1] This is the long version of a paper accepted at AI 2015.

(and UCQs), and letters ϕ, ψ for arbitrary queries. In SPARQL, a CQ of the form $\exists y.A(x) \wedge R(x, y)$ is written as `SELECT ?x WHERE { ?x a A AND ?x :R ?y }`. We use the RDF notation in the SPARQL code, otherwise we use the DL notation. We use bold face notation for lists of variables, e.g., $\boldsymbol{x} = x_1, \ldots, x_n$, and so $\exists \boldsymbol{x}$ is an abbreviation for $\exists x_1, \ldots, \exists x_n$.

A union of CQs (UCQ) is a disjunction of CQs α_i each with the same set of free variables. For a vector of constants $\boldsymbol{a}$ of length n and a formula ϕ with free variables $\boldsymbol{x}$ of the same length, $\phi(\boldsymbol{a})$ is the Boolean query resulting from substituting a_i for x_i, $i \in [n]$. The set of answers for a query $\phi(\boldsymbol{x})$ in interpretation $\mathcal{I}$ is $ans(\phi(\boldsymbol{x}), \mathcal{I}) = \{\boldsymbol{a} = (a_1, \ldots, a_n) \mid a_i$ a constant in Sig and $\mathcal{I} \models \phi(\boldsymbol{a})\}$. The *certain answers* w.r.t. an ontology $\langle \mathcal{A}, \mathcal{T} \rangle$ are: $cert(\phi(\boldsymbol{x}), \langle \mathcal{A}, \mathcal{T} \rangle) = \bigcap_{\mathcal{I} \models \langle \mathcal{A}, \mathcal{T} \rangle} ans(\phi, \mathcal{I})$. For inconsistent ontologies $\boldsymbol{x}$ is bound to all possible bindings $\boldsymbol{a}$ over all constants in the ontology.

For every ABox $\mathcal{A}$ let $DB(\mathcal{A})$ be its minimal Herbrand model. Let QL_1 and QL_2 be query languages over the same signature and OL be an ontology language. QL_1 allows for QL_2-rewriting of query answering w.r.t. OL iff for all queries ϕ in QL_1 and TBoxes $\mathcal{T}$ in OL there exists a query $\phi_{\mathcal{T}}$ in QL_2 such that for all ABoxes $\mathcal{A}$ it holds that: $cert(\phi, \langle \mathcal{A}, \mathcal{T} \rangle) = ans(\phi_{\mathcal{T}}, DB(\mathcal{A}))$. A particularly interesting case is $QL_2 =$ first-order logic (FOL) queries. A well-known fact [8] is: UCQs are FOL rewritable w.r.t. DL-Lite ontologies. Next to rewriting, unfolding has constraints on the used languages in OBDA, too. The backend DBs are in most cases relational database/stream management systems equipped with SQL like query languages, which fulfill the property of domain independence.

A query ϕ is *domain independent* iff for all interpretations $\mathcal{I}, \mathcal{J}$ such that $\mathcal{I}$ is a substructure of $\mathcal{J}$: $ans(\phi, \mathcal{I}) = ans(\phi, \mathcal{J})$ [3].

A flow of time $(T, \leqslant)$ consists of a set T of time points and a linear, transitive, reflexive, and antisymmetric relation $\leqslant$ on T. A *temporal ABox* is a finite set of timestamped ABox axioms $ax\langle t \rangle$, with $t \in T$. We call structures of the form $\langle (\mathcal{A}_i)_{i \in [n]}, \mathcal{T} \rangle$ consisting of a finite sequence of ABoxes and a pure TBox a *sequenced ontology (SO)*. The index i of the ABox $\mathcal{A}_i$ is called its *state*.

A *stream* is a sequence of elements of the form $d\langle t \rangle$ where d is an element from the *stream domain* D and t is a timestamp from a *flow of time*. We assume that the time ordering respects the arrival ordering in the sequence. That is, if $d_1\langle t_1 \rangle$ occurs before $d_2\langle t_2 \rangle$ in the stream, then $t_1 \leqslant t_2$.

The kinds of domains D that we are going to consider are in the first place ABox assertions, but for the semantics of STARQL we will have to consider also different domains.

3 The STARQL Framework

The following example illustrates the main constructors of STARQL. A rational agent has different sensors, in particular different temperatures attached to different components. The agent receives high-level messages within an input stream `Sin`. The agent has some background knowledge on the sensors. The agent has to recognize critical temperatures: Mark every 10 seconds all temperature sensors as critical such that the following holds: In the last 5 minutes there was a monotonic increase on some interval followed by an alert message of category A. A STARQL formalization is given in the following listing.

```
1  CREATE STREAM Sout AS
2  CONSTRUCT  GRAPH NOW { ?s a inCriticalState }
3  FROM Sin[NOW-5min, NOW]->10s, <http://ABox>, <http://TBox>
4  USING PULSE AS START = 0s, FREQUENCY = 10s
5  WHERE { ?s a TempSens }
6  SEQUENCE BY StdSeq AS SEQ
7  HAVING
8  EXISTS i1, i2, i3 IN SEQ
9   0 < i1 AND i2 < MAX AND plus(i2,1,i3) AND
10  GRAPH i3 { ?s :message ?m  . ?m a A-Message } AND
11  FORALL i, j in SEQ FORALL ?x,?y:
12    IF  i1 <= i  AND i<= j  AND j <= i2  AND
13        GRAPH i { ?s :val ?x }  AND GRAPH j { ?s :val ?y }
14    THEN ?x <= ?y
```

The `CONSTRUCT` operator fixes the format of the output stream. The window operator `[NOW-5min, NOW]->10s` gives snapshots of the stream with the frequency of 10s and range of 5 minutes. The `WHERE` clause (l. 5) determines the sensors $?s$.

Already here the agent has to incorporate his background knowledge from the TBox: In order to get all temperature sensors $?s$ he as also to find all type-X sensors.

For every binding of $?s$, the query evaluates conditions that are specified in the `HAVING` clause (l. 7–14). A sequencing method (here `StdSeq`) maps an input stream to a sequence of ABoxes (annotated by states i, j) according to a grouping criterion. Testing for conditions at a state is done with the SPARQL sub-graph mechanism. So, e.g., `GRAPH i3 {?s :message ?m . ?m a A-Message}` (l. 10) asks whether `?s` showed a message of Type A at state `i3`. Here, `i3` is the successor of the end state `i2` in the interval `[i1, i2]` for which one tests monotonicity (`FORALL` condition, l. 12–14).

The grammar of STARQL(OL,ECL) (Fig. 1) contains parameters that have to be specified in its instantiations, the ontology language OL and the embedded condition language ECL. ECL is a query language referring to the signature of the ontology language. STARQL uses ECL conditions in its `WHERE` and `HAVING` clauses. The adequate instantiation of STARQL(OL, ECL) may vary depending on the requirements of the use case.

In order to define the semantics for instantiations of STARQL(OL,ECL), parameter values for OL and ECL have to fulfill the following conditions: There must be a notion of certain answers of an ECL w.r.t. an ontology.

For ease of exposition we assume that the query specifies only one output sub-graph pattern and that there is exactly one static ABox $\mathcal{A}_{st}$ and one TBox $\mathcal{T}$. Similar to the approach of [6], the TBox is assumed to be non-temporal in the sense that there are no special temporal or stream constructors. We specify the denotation $[\![S_{out}]\!]$ of S_{out} recursively by defining the denotations of the

starql	$\longrightarrow$	$[\mathit{prefix}]\ \mathit{createExp}$
$\mathit{createExp}$	$\longrightarrow$	$\texttt{CREATE STREAM}\ \mathit{sName}\ \texttt{AS}\ \mathit{constrExp}$
$\mathit{pulseExp}$	$\longrightarrow$	$\texttt{PULSE WITH}$ $\texttt{START = }\mathit{start},$ $\texttt{FREQUENCY = }\mathit{freq}$
$\mathit{constrExp}$	$\longrightarrow$	$\texttt{CONSTRUCT}\ \mathit{cHead}(\boldsymbol{x},\boldsymbol{y})$ $\texttt{FROM}\ \mathit{listWStrExp}$ $[\texttt{,}\ \underline{\mathit{URIsToA/TBoxes}}]$ $[\texttt{USING}\ \mathit{pulseExp}]$ $\texttt{WHERE}\ \mathit{whereCl}(\boldsymbol{x})$ $\texttt{SEQUENCE BY}\ \mathit{seqMeth}$ $\texttt{HAVING}\ \mathit{safeHCl}(\boldsymbol{x},\boldsymbol{y})$
$\mathit{cHead}(\boldsymbol{x},\boldsymbol{y})$	$\longrightarrow$	$\texttt{GRAPH}\ \mathit{timeExp}\ \mathit{triple}(\boldsymbol{x},\boldsymbol{y})$ $\{\ .\ \mathit{cHead}(\boldsymbol{x},\boldsymbol{y})\}$
$\mathit{listWStrExp}$	$\longrightarrow$	$(\mathit{sName}\mid\mathit{constrExp})\ \mathit{winExp}$ $[\texttt{,}\ \mathit{listWStrExp}]$
winExp	$\longrightarrow$	$\texttt{[}\mathit{timeExp_1},\mathit{timeExp_2}\texttt{]->}\mathit{sl}$
$\mathit{timeExp}$	$\longrightarrow$	$\texttt{NOW}\mid\texttt{NOW -}\ \mathit{constant}$
$\mathit{whereCl}(\boldsymbol{x})$	$\longrightarrow$	$\boxed{\mathit{ECL}(\boldsymbol{x})}$
$\mathit{seqMeth}$	$\longrightarrow$	$\texttt{StdSeq}\mid\mathit{seqMeth}(\sim)$
$\mathit{term}(i)$	$\longrightarrow$	i
$\mathit{term}()$	$\longrightarrow$	$\texttt{MAX}\mid\texttt{0}\mid\texttt{1}$
$\mathit{arAt}(i_1,i_2)$	$\longrightarrow$	$\mathit{term}_1(i_1)\ \mathit{op}\ \mathit{term}_2(i_2)$ $(\mathit{op}\in\{\texttt{<},\texttt{<=},\ \texttt{=},\ \texttt{>},\ \texttt{>=}\})$
$\mathit{arAt}(i_1,i_2,i_3)$	$\longrightarrow$	$\texttt{plus(}\mathit{term}_1(i_1)\texttt{,}$ $\mathit{term}_2(i_2)\texttt{,}$ $\mathit{term}_3(i_3)\texttt{)}$
$\mathit{stateAt}(\boldsymbol{x},i)$	$\longrightarrow$	$\texttt{GRAPH i}\ \boxed{\mathit{ECL}}\,(\boldsymbol{x})$
$\mathit{atom}(\boldsymbol{x})$	$\longrightarrow$	$\mathit{arAt}(\boldsymbol{x})\mid\mathit{stateAt}(\boldsymbol{x})$
$\mathit{hCl}(\boldsymbol{x})$	$\longrightarrow$	$\mathit{atom}(x)\mid\mathit{hCl}(\boldsymbol{x})\ \texttt{OR}\ \mathit{hCl}(\boldsymbol{x})$
$\mathit{hCl}(\boldsymbol{x},\boldsymbol{y})$	$\longrightarrow$	$\mathit{hCl}(\boldsymbol{x})\ \texttt{AND}\ \mathit{hCl}(\boldsymbol{y})$
$\mathit{hCl}(\boldsymbol{x})$	$\longrightarrow$	$\mathit{hCl}(\boldsymbol{x})\ \texttt{AND FORALL}\ \boldsymbol{y}$ $\texttt{IF}\ \mathit{hCl}(\boldsymbol{x},\boldsymbol{y})\ \texttt{THEN}\ \mathit{hCl}(\boldsymbol{x},\boldsymbol{y})$
$\mathit{hCl}(\boldsymbol{x},\boldsymbol{z})$	$\longrightarrow$	$\texttt{EXISTS}\ \boldsymbol{y}\ \mathit{hCl}(\boldsymbol{x},\boldsymbol{y})\ \texttt{AND}$ $\mathit{hCl}(\boldsymbol{z},\boldsymbol{y})$
$\mathit{safeHCl}(\boldsymbol{x})$	$\longrightarrow$	$\mathit{hCl}(\boldsymbol{x})$ ($\boldsymbol{x}$ contains no $\texttt{i}$ variable)

Fig. 1. Syntax for STARQL(<u>OL</u>, $\boxed{\text{ECL}}$) template.

components.

$$
\begin{aligned}
S_{out} = {} & \texttt{CONSTRUCT GRAPH}\ \mathit{timeExpCons}\ \Theta(\boldsymbol{x},\boldsymbol{y}) \\
& \texttt{FROM}\ S_1\ \mathit{winExp_1},\ldots,S_m\ \mathit{winExp_m},\mathcal{A}_{st},\mathcal{T} \\
& \texttt{WHERE}\ \psi(\boldsymbol{x})\ \texttt{SEQUENCE BY}\ \mathit{seqMeth}\ \texttt{HAVING}\ \phi(\boldsymbol{x},\boldsymbol{y})
\end{aligned}
$$

Windowing. Let $[\![S_i]\!]$ for $i \in [m]$ be the streams of timestamped ABox assertions. The denotation of the windowed stream $ws_i = S_i\ \texttt{[}\mathit{timeExp}^i_1, \mathit{timeExp}^i_2\texttt{]->}sl_i$ is defined by specifying a function F^{winExp_i} s.t.: $[\![ws_i]\!] = F^{winExp_i}([\![S_i]\!])$.

$[\![ws_i]\!]$ is a stream with timestamps from the set $T' \subseteq T$, where $T' = (t_j)_{j\in\mathbb{N}}$ is fixed by the pulse declaration with t_0 being the starting time point of the pulse. The domain D of the resulting stream are temporal ABoxes.

Assume that $\lambda t.g^i_1(t) = [\![\mathit{timeExp}^i_1]\!]$ and $\lambda t.g^i_2(t) = [\![\mathit{timeExp}^i_2]\!]$ are the unary functions of time denoted by the time expressions in the window. We have to define for every t_j the temporal ABox $\tilde{\mathcal{A}}^i_{t_j}\langle t_j\rangle \in [\![ws_i]\!]$. If $t_j < sl - 1$, then $\tilde{\mathcal{A}}^i_{t_j} = \emptyset$. Else set first $t^i_{start} = \lfloor t_j/sl \rfloor \times sl$ and $t^i_{end} = max\{t_{start} - (g^i_2(t_j) - g^i_1(t_i)), 0\}$, and define on that basis $\tilde{\mathcal{A}}^i_{t_j} = \{ax\langle t\rangle \mid ax\langle t\rangle \in [\![S]\!]$ and $t^i_{end} \leqslant t \leqslant t^i_{start}\}$. Now, the denotations of all windowed streams are joined w.r.t. the timestamps in T': $js([\![ws_1]\!],\ldots,[\![ws_m]\!]) := \{\big(\bigcup_{i\in[m]} \tilde{\mathcal{A}}^i_t\big)\langle t\rangle \mid t \in T'$ and $\tilde{\mathcal{A}}^i_t\langle t\rangle \in [\![ws_j]\!]\}$.

Sequencing. The stream $js([\![ws_1]\!],\ldots,[\![ws_m]\!])$ is processed according to the sequencing method specified in the query. The output stream has timestamps from T'. The stream domain D now consists of finite sequences of pure ABoxes.

The sequencing methods used in STARQL refer to an equivalence relation $\sim$ to specify which assertions go into the same ABox. The relation $\sim$ is required to respect the time ordering, i.e., it has to be a congruence over T. The equivalence classes are referred to as states and are denoted by variables i, j etc.

Let $\tilde{\mathcal{A}}_t\langle t\rangle$ be the temporal ABox of $jstr$ at t. Let $T'' = \{t_1, \ldots, t_l\}$ be the time points occurring in $\tilde{\mathcal{A}}_t$ and let k' the number of equivalence classes generated by the time points in T''. Then define the sequence at t as $(\mathcal{A}_0, \ldots, \mathcal{A}_{k'})$ where for every $i \in [k']$ the ABox $\mathcal{A}_i$ is $\mathcal{A}_i = \{ax\langle t'\rangle \mid ax\langle t'\rangle \in \tilde{\mathcal{A}}_t$ and t' in i^{th} equiv. class$\}$. The standard sequencing method `StdSeq` is just $seqMeth(=)$. Let $F^{seqMeth}$ be the function realizing the sequencing.

WHERE Clause. In the `WHERE` clause only $\mathcal{A}_{st}$ and $\mathcal{T}$ are relevant for the answers. So, purely static conditions (e.g. asking for sensor types as in the example above) are evaluated only on $\mathcal{A}_{st} \cup \mathcal{T}$. The result are bindings $\boldsymbol{a}_{wh} \in cert(\psi(\boldsymbol{x}), \langle \mathcal{A}_{st}, \mathcal{T}\rangle)$. This set of bindings is applied to the `HAVING` clause $\phi(\boldsymbol{x}, \boldsymbol{y})$.

HAVING Clause. STARQL's semantics for the `HAVING` clauses relies on the certain answer semantics of the embedded ECL conditions. We have to define the semantics of $\phi(\boldsymbol{a}_{wh}, \boldsymbol{y})$ for every binding $\boldsymbol{a}_{wh}$ from the evaluation of the `WHERE` clause. We declare for every t how to get bindings for $\boldsymbol{y}$. Assume that the sequence of ABoxes at t is $seq = (\mathcal{A}_0, \ldots, \mathcal{A}_k)$. We define the *separation-based* certain answers: $cert_{sep}(\phi(\boldsymbol{a}_{wh}, \boldsymbol{y}), \langle \mathcal{A}_i \cup \mathcal{A}_{st}, \mathcal{T}\rangle)$. If for any i the pure ontology $\langle \mathcal{A}_i \cup \mathcal{A}_{st}, \mathcal{T}\rangle$ is inconsistent, then we set $cert_{sep} =$ NIL, where NIL is a new constant not contained in the signature. In the other case, the bindings are defined as follows. For t one constructs a sorted first order logic structure $\mathcal{I}_t$: The domain of $\mathcal{I}_t$ consists of the index set $\{0, \ldots, k\}$ as well as the set of all individual constants of the signature. For every state atom *stateAt* `GRAPH i` $ECL(\boldsymbol{z})$ in $\phi(\boldsymbol{a}_{wh}, \boldsymbol{y})$ with free variables $\boldsymbol{z}$ having length l, say, introduce an $(l+1)$-ary symbol R and replace `GRAPH i` $ECL(\boldsymbol{z})$ by $R(\boldsymbol{z}, i)$. The denotation of R in $\mathcal{I}_t$ is then defined as the set of certain answers of the embedded condition $ECL(\boldsymbol{z})$ w.r.t. the i^{th} ABox $\mathcal{A}_i$: $R^{\mathcal{I}_t} = \{(\boldsymbol{b}, i) \mid \boldsymbol{b} \in cert(ECL(\boldsymbol{z}), \langle \mathcal{A}_i \cup \mathcal{A}_{st}, \mathcal{T}\rangle)\}$. Constants denote themselves in $\mathcal{I}_t$. This fixes a structure $\mathcal{I}_t$ with finite denotations of its relation symbols. The evaluation of the `HAVING` clause is then nothing more than evaluating the FOL formula (after substitutions) on the structure $\mathcal{I}_t$.

Let $F^{\phi(\boldsymbol{a}_{wh}, \boldsymbol{y})}$ be the function that maps a stream of ABox sequences to the set of bindings $(\boldsymbol{b}, t)$ where $\boldsymbol{b}$ is the binding for $\phi(\boldsymbol{a}_{wh}, \boldsymbol{y})$ at time point t.

Summing up, we get the following denotational decomposition:

$$\begin{aligned}\llbracket S_{out}\rrbracket = \{&\texttt{GRAPH}\ \llbracket timeExpCons\rrbracket\ \Theta(\boldsymbol{a_{wh}}, \boldsymbol{b}) \mid \boldsymbol{a_{wh}} \in cert(\psi(\boldsymbol{x}), \mathcal{A}_{st} \cup \mathcal{T}) \text{ and} \\ &(\boldsymbol{b}, t) \in F^{\phi(\boldsymbol{a_{wh}}, \boldsymbol{y})}\big(F^{seqMeth}(js(F^{winExp_1}(\llbracket S_1\rrbracket), \ldots, F^{winExp_m}(\llbracket S_m\rrbracket)))\big)\}\end{aligned}$$

Properties. The motivation for such a definition of the semantics of `HAVING` clauses is a strict separation of the semantics provided by the embedded condition languages ECL and the semantics used on top of it. This allows for embedding any ECL without repeatedly redefining the semantics. The separation-based semantics has an immediate consequence for perfect rewritability, which is adapted to the sequenced setting as follows. Let $\mathcal{O} = \langle (\mathcal{A}_i)_{i \in [n]}, \mathcal{T}\rangle$ be a SO. Let the canonical model $DB((\mathcal{A}_i)_{i \in [n]})$ of a sequence of ABoxes be defined as the sequence of minimal Herbrand models $DB(\mathcal{A}_i)$ of the ABoxes $\mathcal{A}_i$. Let QL_1 and QL_2 be two query languages over the same signature of a SO and OL be a language for the sequenced ontologies SO.

Definition 1. *QL_1 allows for QL_2-rewriting of query answering w.r.t. the ontology language OL iff for all queries ϕ in QL_1 and TBoxes $\mathcal{T}$ in OL there exists a query $\phi_{\mathcal{T}}$ in QL_2 such that for all $n \in \mathbb{N}$ and all sequences of ABoxes $(\mathcal{A}_i)_{i\in[n]}$ it holds that: $cert(\phi, \langle(\mathcal{A}_i)_{i\in[n]}, \mathcal{T}\rangle) = ans(\phi_{\mathcal{T}}, DB((\mathcal{A}_i)_{i\in[n]}))$*

We call such an ECL language *rewritability closed* w.r.t. OL iff ECL allows for perfect rewriting s.t. the rewritten formula is again an ECL condition.

Proposition 1. *Let ECL be a rewritability-closed condition language and consider the instantiation of `HAVING` called QL_1. Then QL_1 allows for QL_1 rewriting for separation-based certain query answering w.r.t. OL.*

4 Separation-Based versus Holistic Semantics

We define a new, *holistic* semantics STARQL `HAVING` clauses and show that for a fragment of the `HAVING` clauses we get the same answers as w.r.t. the separation-based semantics. This allows to compare STARQL's expressivity with temporal-logic oriented query languages. In particular, we show that the `HAVING` fragment captures (a safe fragment) of the LTL inspired query language of temporal conjunctive queries (TCQs) [6]. We assume that we have a `HAVING` clause where the free variables $\boldsymbol{x}$ of the `WHERE` clause are already bound. With every time point t an SO $\langle(\mathcal{A}_i)_{i\in[n_t]}, \mathcal{T}\rangle$ is associated. The length of the sequence n_t may depend on the time point t. As we now fix it, we write just n for n_t.

Definition 2. *Let $\mathcal{O} = \langle(\mathcal{A}_i)_{i\in[n_t]}, \mathcal{T}\rangle$ be a SO. Let $\hat{\mathcal{I}} = (\mathcal{I}_i)_{0\leqslant i\leqslant n}$ be a sequence of interpretations $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$ over a fixed non-empty domain Δ where the constants' interpretations do not change for different $\mathcal{I}_i, \mathcal{I}_j$. Then $\hat{\mathcal{I}}$ is a model of $\mathcal{O}$ (written $\hat{\mathcal{I}} \models \mathcal{O}$) if $\mathcal{I}_i \models \langle\mathcal{A}_i, \mathcal{T}\rangle$ for all $i \in [n]$.*

All interpretations $\mathcal{I}_i$ have the same domain Δ. The constants' denotations does not change from state to state, hence they are considered rigid. The TBox $\mathcal{T}$ is assumed to be a non-temporal TBox. Its inclusions hold at every time point.

We see that the set of models of $\mathcal{O} = \langle(A_i)_{i\in[n]}, \mathcal{T}\rangle$ is just the cartesian product of all sets of models $\mathcal{I}_i$ of the local ontologies $\mathcal{T} \cup \mathcal{A}_i$.

Proposition 2. $\{\hat{\mathcal{I}} \mid \hat{\mathcal{I}} \models \langle(A_i)_{i\in[n]}, \mathcal{T}\rangle\} = \bigtimes_{i\in[n]}\{\mathcal{I} \mid \mathcal{I} \models \langle A_i, \mathcal{T}\rangle\}$

The satisfaction relation between sequences of interpretations Boolean `HAVING` clauses is defined as follows.

Definition 3. *Let $\hat{\mathcal{I}} = (\mathcal{I}_i)_{i\in[n]}$ be a sequence of interpretations and σ be an assignment of individuals $d \in \Delta$ to individual variables and numbers $i \in [n]$ to* *state variables i, j. Let $\sigma[j \mapsto \underline{j}]$ be the variant of σ where j is assigned $\underline{j}$. The semantics are defined as follows:*

$$
\begin{aligned}
\hat{\mathcal{I}}, \sigma \models \texttt{GRAPH } i\ \alpha \ &\textit{iff}\ \mathcal{I}_{\sigma(i)} \models \alpha \\
\hat{\mathcal{I}}, \sigma \models \texttt{EXISTS } i\ \phi \ &\textit{iff there is}\ \underline{i} \in [n]\ \textit{s.t.}\ \hat{\mathcal{I}}, \sigma[i \mapsto \underline{i}] \models \phi \\
\hat{\mathcal{I}}, \sigma \models \texttt{FORALL } i\ \phi \ &\textit{iff for all}\ \underline{i} \in [n]\ \textit{it holds that}\ \hat{\mathcal{I}}, \sigma[i \mapsto \underline{i}] \models \phi \\
\hat{\mathcal{I}}, \sigma \models \texttt{EXISTS } x\ \phi \ &\textit{iff there is}\ d \in \Delta\ \textit{s.t.}\ \hat{\mathcal{I}}, \sigma[x \mapsto d] \models \phi \\
\hat{\mathcal{I}}, \sigma \models \texttt{FORALL } x\ \phi \ &\textit{iff for all}\ d \in \Delta\ \textit{it holds that}\ \hat{\mathcal{I}}, \sigma[x \mapsto d] \models \phi \\
\hat{\mathcal{I}}, \sigma \models \phi_1\ \texttt{AND}\ \phi_2 \ &\textit{iff}\ \hat{\mathcal{I}}, \sigma \models \phi_1\ \textit{and}\ \hat{\mathcal{I}}, \sigma \models \phi_2 \\
\hat{\mathcal{I}}, \sigma \models \phi_1\ \texttt{OR}\ \phi_2 \ &\textit{iff}\ \hat{\mathcal{I}}, \sigma \models \phi_1\ \textit{or}\ \hat{\mathcal{I}}, \sigma \models \phi_2 \\
\hat{\mathcal{I}}, \sigma \models \phi_{arAt} \ &\textit{iff}\ \mathcal{I}_0, \sigma \models \phi_{arAt} \\
\hat{\mathcal{I}} \models \phi \ &\textit{iff}\ \hat{\mathcal{I}}, \emptyset \models \phi
\end{aligned}
$$

For a `HAVING` clause $\phi(\boldsymbol{x})$ and a sequence of interpretations $\hat{\mathcal{I}}$, the set of answers is $ans(\phi(\boldsymbol{x}), \hat{\mathcal{I}}) = \{\boldsymbol{a} \mid \hat{\mathcal{I}} \models \phi(\boldsymbol{a})\}$. The set of certain answers w.r.t. a SO $\mathcal{O} = \langle (A_i)_{i \in [n]}, \mathcal{T} \rangle$ is: $cert_h(\phi, \mathcal{O}) = \bigcap_{\hat{\mathcal{I}} \models \mathcal{O}} ans(\phi, \hat{\mathcal{I}})$

We consider a fragment of the `HAVING` clauses for OL = DL-Lite and ECL = UCQ and denote it by $\mathcal{L}^{\exists}_{HCL}$: We disallow the operators `FORALL x` and `EXISTS x`. The implicit existential quantifiers in the UCQs are kept. We show that the original separation-based semantics and the holistic certain answer semantics (denoted $cert_h$) are the same on this fragment.

Theorem 1. *For any SO $\mathcal{O} = \langle (A_i)_{i \in [n]}, \mathcal{T} \rangle$ and any $\phi \in \mathcal{L}^{\exists}_{HCL}$ the following equality holds: $cert_h(\phi, \mathcal{O}) = cert_{sep}(\phi, \mathcal{O})$.*

In preparing the proof, we transform, for every $n \in \mathbb{N}$, the formulas in $\mathcal{L}^{\exists}_{HCL}$ into formulas (depending on n) that do not contain any quantifiers over the states of the sequence but may contain constants ι_j (for $j \in [n]$) denoting the states. Denote the rewriting by $\tau^n_{\iota_j}$ with parameters ι_j, n.

$\tau^n_{\iota_j}(\texttt{GRAPH i } \alpha) = \texttt{GRAPH j } \alpha$; $\quad \tau^n_{\iota_j}(\phi_1 \texttt{ AND } \phi_2) = \tau^n_{\iota_j}(\phi_1) \texttt{ AND } \tau^n_{\iota_j}(\phi_2)$
$\tau^n_{\iota_j}(\phi_1 \texttt{ OR } \phi_2) = \tau^n_{\iota_j}(\phi_1) \texttt{ OR } \tau^n_{\iota_j}(\phi_2)$; $\quad \tau^n_{\iota_j}(\phi_{arAt}) = \phi_{arAt}$
$\tau^n_{\iota_j}(\texttt{EXISTS i } \phi) = \tau^n_0(\phi) \texttt{ OR } \tau^n_1(\phi) \texttt{ OR } \dots \texttt{ OR } \tau^n_n(\phi)$
$\tau^n_{\iota_j}(\texttt{FORALL i } \phi) = \tau^n_0(\phi) \texttt{ AND } \tau^n_1(\phi) \texttt{ AND } \dots \texttt{ AND } \tau^n_n(\phi)$
$\tau^n(\phi) = \tau^n_{\iota_n}(\phi)$ (for any ϕ)
So the resulting formula $\tau^n(\phi)$ is equivalent to ϕ and it is made up by atoms of the form `GRAPH` $\iota_j\ \psi$ for ψ being a CQ and ι_j a constant denoting $j \in [n]$.

Proof. Without loss of generality we may assume that on either side of a conjunction the same set of variables occur. A semantic preserving transformation (adding identities) is possible. Let the resulting language be denoted $\mathcal{L}^{\exists,=}_{HCL}$. If $\mathcal{O} = \langle (\mathcal{A}_i)_{i \in [n]}, \mathcal{T} \rangle$ is not consistent, then we assumed that $cert_h(\phi, \mathcal{O}) = \text{NIL} = cert_{sep}(\phi, \mathcal{O})$. So in the following we may assume that $\mathcal{O}$ is consistent which means that for all $i \in [n]$ the local ontologies $\langle \mathcal{A}_i, \mathcal{T} \rangle$ are consistent.

We first verify that for all atoms ϕ that $cert_{sep}(\phi, \mathcal{O}) = cert_h(\phi, \mathcal{O})$ holds. This is trivial if ϕ is not a state atom. If ϕ is a state atom of the form GRAPH ι_i α with a CQ α , then $cert_{sep}(\phi, \mathcal{O}) = cert_{sep}(\phi, \langle \mathcal{A}_i, \mathcal{T} \rangle) = cert(\phi, \langle \mathcal{A}_i, \mathcal{T} \rangle)$. Now $cert(\phi, \langle \mathcal{A}_i, \mathcal{T} \rangle) = cert_h(\phi, \mathcal{O})$ holds due to the definitions of satisfaction relation in the case for state atoms and due to Proposition 2.

Now we show that $cert_h$ can be pushed through to the atoms by showing that it distributes over AND and OR. Let first $\phi = \phi_1$ AND ϕ_2 where the conjuncts have the same set of free variables, say $x_1, \ldots, x_n$. Then the following chain of equalities holds: $cert_h(\phi_1 \text{ AND } \phi_2, \mathcal{O}) = \bigcap_{\hat{\mathcal{I}} \models \mathcal{O}} ans_{wh}(\phi_1 \text{ AND } \phi_2, \hat{\mathcal{I}}) = \bigcap_{\hat{\mathcal{I}} \models \mathcal{O}} ans(\phi_1, \hat{\mathcal{I}}) \cap \bigcap_{\hat{\mathcal{I}} \models \mathcal{O}} ans_{wh}(\phi_2, \hat{\mathcal{I}})$. The latter is $cert_h(\phi_1, \mathcal{O}) \cap cert_h(\phi_2, \mathcal{O})$. Now let $\phi = \phi_1$ OR ϕ_2 with free variables $x_1, \ldots, x_n$. We have $cert_h(\phi_1, \mathcal{O})) \cup cert_h(\phi_2, \mathcal{O})) \subseteq cert_h(\phi_1 \text{ OR } \phi_2, \mathcal{O})$. For the other direction assume that $\boldsymbol{c} \in cert_h(\phi_1 \text{ OR } \phi_2, \mathcal{O})$; assume that $\boldsymbol{c} \notin cert_h(\phi_1, \mathcal{O}) \cup cert_h(\phi_2, \mathcal{O})$. Then there must be models $\hat{\mathcal{I}}_1, \hat{\mathcal{I}}_2 \models \mathcal{O}$ such that $\boldsymbol{c} \notin ans(\phi_1, \hat{\mathcal{I}}_1)$ and $\boldsymbol{c} \notin ans(\phi_2, \hat{\mathcal{I}}_2)$. It follows that $\hat{\mathcal{I}}_1 \models \phi_2(\boldsymbol{c})$ and $\hat{\mathcal{I}}_2 \models \phi_1(\boldsymbol{c})$. Now, $\boldsymbol{c}$ is also in the answer set for the canonical model $can(\mathcal{O})$ [8], i.e., $\boldsymbol{c} \in ans(\phi_1 \text{ OR } \phi_2, can(\mathcal{O}))$, so either $can(\mathcal{O}) \models \phi_1(\boldsymbol{c})$ or $can(\mathcal{O}) \models \phi_2(\boldsymbol{c})$. But as $can(\mathcal{O})$ is a universal model there is a homomorphism into every model of $\mathcal{O}$, in particular for $\hat{\mathcal{I}}_1$ and $\hat{\mathcal{I}}_2$. As homomorphisms preserve positive existential clauses we must have either $\boldsymbol{c} \in ans(\phi_1, \hat{\mathcal{I}}_1)$ or $\boldsymbol{c} \in ans(\phi_2, \hat{\mathcal{I}}_2)$, contradicting our assumption. $\square$

As a corollary we get rewritability for STARQL queries in the holistic semantics.

Theorem 2. *Let QL_1 be the instantiation of the HAVING clause language with ECL = UCQ and OL = DL-Lite. Then QL_1 allows for QL_1 rewriting for holistic certain query answering w.r.t. OL on the ABox sequence.*

Moreover, as HAVING clauses are based on FOL with $<$, plus (the MAX operator can be rewritten with $<$) and the state variables i can be pushed into the UCQs, we get the following additional corollary:

Corollary 1. *Let QL_1 be the instantiation of HAVING clauses with ECL = UCQ and OL = DL-Lite. Then QL_1 allows for FOL($<$, plus) rewriting for holistic certain query answering w.r.t. OL on the ABox sequence.*

These rewritability theorems still do not say wether it is required to generate/materialize the ABox sequence or whether it can be defined as a view. For the standard sequencing method there is a one-to-one correspondence between pairs of state and developing time (i, NOW) and timestamps. Hence, it easily follows that also the sequencing can be compiled into the query.

Corollary 2. *Let QL_1 be the instantiation of HAVING clauses with ECL = UCQ and OL = DL-Lite and assume that the sequence of ABoxes is generated by standard sequencing. Then QL_1 allows for FOL($<$, plus) rewriting for holistic certain query answering w.r.t. OL on the stream of timestamped ABox axioms.*

A consequence of this corollary is: Query answering w.r.t. the intensional knowledge has been reduced to simple query answering on the data, which can be accomplished by relational stream query languages such as CQL [1].

$$\begin{aligned}
&\phi \longrightarrow CQ \mid & \hat{\mathcal{I}}, i \models \exists y_1,\ldots,y_m.\rho &\text{ iff } \mathcal{I}_i \models \exists y_1,\ldots,y_m.\rho\\
&\quad \phi \wedge \phi \mid \phi \vee \phi \mid & \hat{\mathcal{I}}, i \models \phi_1 \wedge \phi_2 &\text{ iff } \hat{\mathcal{I}}, i \models \phi_1 \text{ and } \hat{\mathcal{I}}, i \models \phi_2\\
&\quad \bigcirc\phi \mid \bullet\phi \mid & \hat{\mathcal{I}}, i \models \phi_1 \vee \phi_2 &\text{ iff } \hat{\mathcal{I}}, i \models \phi_1 \text{ or } \hat{\mathcal{I}}, i \models \phi_2\\
&\quad \bigcirc^-\phi \mid \bullet^-\phi \mid & \hat{\mathcal{I}}, i \models \bigcirc\phi_1 &\text{ iff } i < n \text{ and } \hat{\mathcal{I}}, i+1 \models \phi_1\\
&\quad \phi \mathsf{U} \phi \mid \phi \mathsf{S} \phi & \hat{\mathcal{I}}, i \models \bullet\phi_1 &\text{ iff } i < n \text{ implies } \hat{\mathcal{I}}, i+1 \models \phi_1\\
& & \hat{\mathcal{I}}, i \models \bigcirc^-\phi_1 &\text{ iff } i > 0 \text{ and } \hat{\mathcal{I}}, i-1 \models \phi_1\\
& & \hat{\mathcal{I}}, i \models \bullet^-\phi_1 &\text{ iff } i > 0 \text{ implies } \hat{\mathcal{I}}, i-1 \models \phi_1\\
& & \hat{\mathcal{I}}, i \models \phi_1 \mathsf{U} \phi_2 &\text{ iff } \exists k : i \leqslant k \leqslant n, \hat{\mathcal{I}}, k \models \phi_2\\
& & &\quad \text{and } \hat{\mathcal{I}}, j \models \phi_1 \forall j, i \leqslant j < k\\
& & \hat{\mathcal{I}}, i \models \phi_1 \mathsf{S} \phi_2 &\text{ iff } \exists k : 0 \leqslant k \leqslant i, \hat{\mathcal{I}}, k \models \phi_2\\
& & &\quad \text{and } \hat{\mathcal{I}}, j \models \phi_1 \forall j, k < j \leqslant i.
\end{aligned}$$

Fig. 2. TCQs: syntax and semantics

5 Comparison with TCQs

The fragment of the `HAVING` clause language, for which we showed the equivalence of the holistic and the separation-based semantics, is still expressive enough to simulate query languages that combine temporal logic operators with lightweight DL languages such as the language of temporal conjunctive queries (TCQs) [6].

TCQs are defined by following a weak integration of conjunctive queries (CQs) and a linear temporal logic (LTL) template. The syntax is given on the left-hand side in Figure 2.

Definition 4. *Let ϕ be a Boolean TCQ. For $\hat{\mathcal{I}} = (\mathcal{I}_i)_{i \in [n]}$ and $i \in [n]$ one defines $\hat{\mathcal{I}}, i \models \phi$ by induction on the structure of ϕ as in the right-hand side of Figure 2.*

The set of answers at i is defined as $ans(\phi, \hat{\mathcal{I}}, i) = \{\boldsymbol{a} \mid \hat{\mathcal{I}}, i \models \phi(\boldsymbol{a})\}$. For a SO $\mathcal{O}$ the set of certain answers at i is $cert(\phi, \mathcal{O}) = \bigcap_{\hat{\mathcal{I}} \models \mathcal{I}} ans(\phi, \hat{\mathcal{I}}, i)$. The set of (certain) answers is defined as the (certain) answers at the last state: $ans(\phi, \hat{\mathcal{I}}) = ans(\phi, \hat{\mathcal{I}}, n)$ and $cert(\phi, \mathcal{O}) := cert(\phi, \mathcal{O}, n)$.

A simple example shows that TCQs (with natural semantics) is not domain independent—though it is intended to be used in the OBDA paradigm: Take $\phi(x, y) = A(x) \vee B(y)$ and consider the interpretations $\mathcal{I} = (\{a\}, \cdot^{\mathcal{I}}), \mathcal{J} = (\{a, b\}, \cdot^{\mathcal{J}})$ with $A^{\mathcal{I}} = A^{\mathcal{J}} = \{a\}$ and $B^{\mathcal{I}} = B^{\mathcal{J}} = \emptyset$: It is $(a, b) \in ans(\phi, \mathcal{J})$ but $(a, b) \notin ans(\phi, \mathcal{I})$. As the TCQs are not domain independent, we consider the following safe fragment TCQ^s where the rule for $\vee$ is replaced by: If ϕ_1, ϕ_2 are in TCQ^s and have the same free variables, then $\phi_1 \vee \phi_2$ is in TCQ^s; $\bullet$, $\bullet^-$ are disallowed; the rules for S, U are replaced by: If ϕ_1, ϕ_2 are in TCQ^s and have the same free variables, then $\phi_1 \mathsf{U} \phi_2$ and $\phi_1 \mathsf{S} \phi_2$ are in TCQ^s. TCQ^{s}s are embeddable into STARQL. The translation θ is the usual one known from translating modal logics into predicate logic. It just simulates the semantics for TCQs within the object language of STARQL. For example, $\theta_i(\exists y_1, \ldots, y_m.\psi) =$ `GRAPH i` ψ;

$\theta_i(\phi_1 \vee \phi_2) = \theta_i(\phi_1)$ `OR` $\theta_i(\phi_2)$; $\theta_i(\bigcirc^- \phi_1) = i > 0$ `AND` $\theta_{i-1}(\phi_1)$. From the similarity of the semantics for TCQ^s and the holistic semantics of $\mathcal{L}^{\exists}_{HCL}$-queries we see that the transformation θ yields for every TCQ^s ϕ a $\mathcal{L}^{\exists}_{HCL}$-query with the same set of certain answers.

Proposition 3. *For all SOs $\mathcal{O}$ and TCQ^s ϕ: $cert(\phi, \mathcal{O}) = cert_h(\theta(\phi), \mathcal{O})$*

Due to some unsafe operators in TCQs, STARQL embeds not all TCQs. This is an intended feature of STARQL as it was intended to be applicable for strict OBDA scenarios with safe query languages s.a. SQL. The full language of TCQs allows for queries for which an implementation in a domain independent language such as SQL leads to performance issues: In the case of arbitrary disjunctions a function has to be implemented that returns all constants in the active domain. On the other hand, STARQL is more expressive than the safe fragment of TCQs: it offers (different) means to generate ABox sequences, whereas for TCQs it is assumed that these are given in advance. Moreover, TCQs allow for quantifiers only within embedded CQs, but cannot handle outer quantification, which is needed in order to, e.g., express the monotonicity condition as in the example from the beginning.

6 Conclusion

The STARQL query language framework is part of the recent venture of adapting OBDA for stream-temporal reasoning and as such is a candidate for performant high-level stream processing within rational agents. In this paper we extended the picture on the different approaches by positioning STARQL w.r.t. the language of TCQs, which was possible by defining a temporal-logic oriented semantics for STARQL in addition to the original semantics.

References

1. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. The VLDB Journal 15, 121–142 (2006)
2. Artale, A., Kontchakov, R., Wolter, F., Zakharyaschev, M.: Temporal description logic for ontology-based data access. In: IJCAI 2013. pp. 711–717 (2013)
3. Avron, A.: Constructibility and decidability versus domain independence and absoluteness. Theor. Comput. Sci. 394(3), 144–158 (Apr 2008)
4. Baader, F., Bauer, A., Baumgartner, P., Cregan, A., Gabaldon, A., Ji, K., Lee, K., Rajaratnam, D., Schwitter, R.: A novel architecture for situation awareness systems. In: Giese, M., Waaler, A. (eds.) Proceedings of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux 2009). Lecture Notes in Computer Science, vol. 5607, pp. 77–92. Springer-Verlag (2009)
5. Baader, F., Borgwardt, S., Lippmann, M.: Temporalizing ontology-based data access. In: CADE-13 (2013)
6. Borgwardt, S., Lippmann, M., Thost, V.: Temporal query answering in the description logic DL-Lite. In: FroCos 2013. LNCS, vol. 8152, pp. 165–180 (2013)
7. Calbimonte, J.P., Jeung, H., Corcho, O., Aberer, K.: Enabling query technologies for the semantic sensor web. Int. J. Semant. Web Inf. Syst. 8(1), 43–63 (Jan 2012)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and databases: The DL-Lite approach. In: 5th Int. Reasoning Web Summer School (RW 2009), LNCS, vol. 5689, pp. 255–356. Springer (2009)
9. Della Valle, E., Ceri, S., Barbieri, D., Braga, D., Campi, A.: A first step towards stream reasoning. In: Future Internet – FIS 2008, LNCS, vol. 5468, pp. 72–81. Springer (2009)
10. Heintz, F., Kvarnström, J., Doherty, P.: Bridging the sense-reasoning gap: Dyknow - stream-based middleware for knowledge processing. Advanced Engineering Informatics 24(1), 14–26 (2010)
11. Özçep, Ö.L., Möller, R., Neuenstadt, C.: A stream-temporal query language for ontology based data access. In: KI 2014: Advances in Artificial Intelligence. LNCS, vol. 8736, pp. 183–194. Springer (2014)
12. Phuoc, D.L., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: International Semantic Web Conference (1). pp. 370–388 (2011)
13. Russell, S.J., Norvig, P.: Artificial Intelligence – A Modern Approach. Prentice Hall (1995)