CQELS 2.0: Towards A Unified Framework for Semantic Stream Fusion

Anh Le-Tuan, Manh Nguyen-Duc, Chien-Quang Le, Trung-Kien Tran,
Manfred Hauswirth, Thomas Eiter and Danh Le-Phuoc

Abstract We present CQELS 2.0, the second version of **C**ontinuous **Q**uery **E**valuation over **L**inked **S**treams. CQELS 2.0 is a platform-agnostic federated execution framework towards semantic stream fusion. In this version, we introduce a novel neural-symbolic stream reasoning component that enables specifying deep neural network (DNN) based data fusion pipelines via logic rules with learnable probabilistic degrees as weights. As a platform-agnostic framework, CQELS 2.0 can be implemented for devices with different hardware architectures (from embedded devices to cloud infrastructures). Moreover, this version also includes an adaptive federator that allows CQELS instances on different nodes in a network to coordinate their resources to distribute processing pipelines by delegating partial workloads to their peers via sub-scribing continuous queries.

1 Introduction and Motivation

CQELS was proposed as one of the first RDF stream processing engines to solve the data integration problem for sensor data, Internet of Things, and the Web of data [14].

Anh Le-Tuan · Manh Nguyen-Duc
Open Distirbuted Systems, Technical University of Berlin, Germany

Chien-Quang Le
University of Science, Hue, Vietnam

Trung-Kien Tran
Bosch Center for Artificial Intelligence, Renningen, Germany

Thomas Eiter
Vienna University of Technology

Manfred Hauswirth · Danh Le-Phuoc
Open Distirbuted Systems, Technical University of Berlin
Fraunhofer Institute for Open Communication Systems, Berlin, Germany

The key feature of this data integration approach is the semantic interoperabiliy enabled by RDF data models. The framework has been used in various industry and research applications both as open source and commercial software.

The recent advances in *machine learning* (ML) with *deep neural network* (DNN) leads to the enormous amount of data in various formats, and in many cases from multimodal stream data with high accuracy. Therefore, the existing framework is no longer suitable for the new application scenarios and requires significant extensions. Towards this goal, in CQELS 2.0, we integrate a number of new stream data types such as video streams, LiDARs, and support more hardwares such as ARM and mobiles [17, 20]. Moreover, we provide data fusion operations in our engines [15, 21]. For example, the operation for object detection DNNs [22, 24] returns a set of bounding boxes and object classes given an image or video frames. Thus, CQELS 2.0 is designed as a DNN-based data stream fusion framework. Along with the new feature for different data sources, the CQELS-QL query language has evolved from supporting event query patterns [6, 8] to probabilistic reasoning in [15, 21].

To implement such complex features and also to support for future demand in research, we bring the existing CQELS framework to the next level with new architecture and design. The new design is driven by requirements from use cases and research problems from various research projects (e.g. DFG COSMO[1] and BMBF BIFOLD[2]) and industry partners. For example, application scenarios in edge intelligence [26, 9] and industry 4.0 from DellEMC, Siemens and Bossh motivate us to build *autonomous processing kernels* powered by CQELS [20, 21]. Such systems deal with city-scale camera deployments which have become ubiquitous in smart cities and traffic monitoring with a continuously increasing in size and utilization of their deployments. For instance, the British Security Industry Association estimates that there are between 4 to 5.9 millions CCTV cameras in the UK [19]. To this end, the new design of CQELS framework has to deal with not only the new level of complexity of data sources, processing operations and deployment settings, but also with significantly larger processing scale in terms of throughput and volume.

2 CQELS 2.0 Framework for Semantic Stream Fusion

Semantic stream reasoning (SSR) [15] enables the data fusion of multimodal stream data such as camera and LiDARs via declarative rules. Such rules can be written in SPARQL-based or Answer Set Programming (ASP) syntax. And the stream data flow between data fusion operations can be represented as standardized data formats, e.g, RDF Star. SSR generalizes the data model and processing operations of the previous CQELS engines, hence, CQELS 2.0 framework uses SSR formalisation and abstractions to facilitate the architecture design and implementation. Figure 1 illustrates the conceptual design of CQELS 2.0 framework that comprises several

[1] https://gepris.dfg.de/gepris/projekt/453130567?language=en

[2] https://bifold.berlin/

components. The Feature Extractor component extracts the features of interest from the incoming information streams and maps them to the symbolic representation. The feature data is described with a *neuro-symbolic stream model* [15] and its semantic is enriched via linked knowledge graphs.

The Reasoning Programs Producer component takes responsibility to generate *semantic stream reasoning programs* [15] which specify the fusion pipeline and the decision logic to choose the most likely state of the world at each evaluation. The reasoning program is evaluated by the Reasoner component, which employs an ASP solver. There are two types of reasoning rules: *hard rules* and *soft rules*. The hard rule is used for background knowledge given by (non-monotonic) common-sense and domain knowledge that is regarded as "always true". The soft rules expresses association hypotheses with weights corresponding to probability degrees of these rules. The weights of the rules are determined by the Learning Agent component in the starting phase.

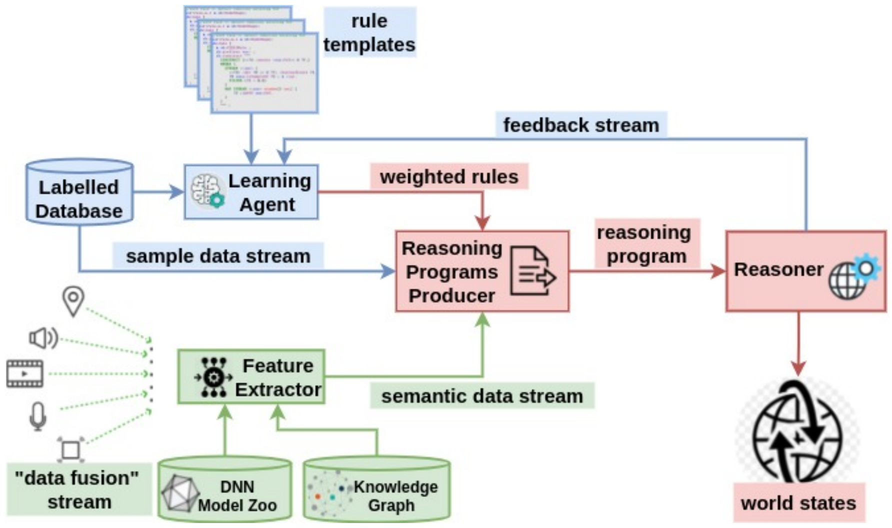


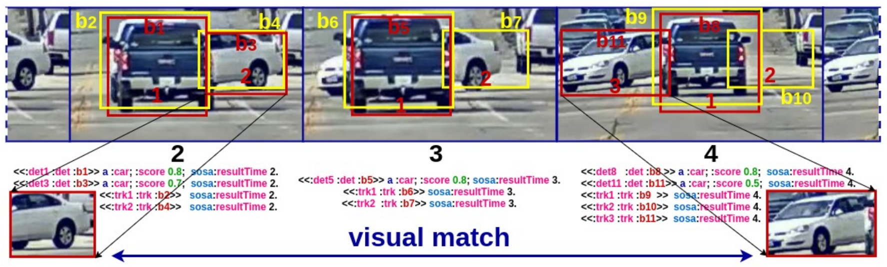
Fig. 1 The overview of conceptual design of CQELS 2.0

The workflow of the framework is as follows. For the setup, the symbolic training samples are constructed from the labeled data and are fed to the Learning Agent component. The Learning Agent component computes a vector of weights for the soft rules which are stored as rule templates and passes them to the Reasoning Programs Producer component. For each training iteration, a *feedback stream* returns the reasoning results back to Learning Agent component. The Learning Agent component adjusts the weights of the soft rules until the answer sets (returned as feedback streams) describe the most likely ground truth.

3 Provisional Features

In this section, we present details of a provisional feature of CQELS 2.0 in the *multiple object tracking* (MOT) scenario [2] which is normally programmed in C/C++ or Python. We will show that with CQELS 2.0, the MOT can be implemented in a declarative fashion with rules and queries.

Particularly, we focus on the tracking-by-detection approach [5], which is widely used for the MOT problem in computer vision. The key operations in such approach are as following: 1) detection of objects (using DDN-based detector), 2) propagating object states (e.g., location and velocity) into future frames, 3) associating current detection with existing objects, and 4) managing the lifespan of the tracked objects. For example, Figure 2 illustrates the SORT algorithm [3] which is a simple object tracking algorithm based on DNN detectors such as SSD [18] or YOLO [23]. To associate resultant detections with existing targets, SORT uses a Kalman filter [4] to predict the new locations of targets in the current frame. At time point 2, the red boxes b_1 and b_3 are detected by a detector. The yellow boxes b_2 and b_4 are predicted by a Kalman filter based on the boxes of tracklets from the previous frame. Then, the SORT algorithm computes an associative cost matrix between detections and targets based on the intersection-over-union (IOU) distance between each detection and all predicted bounding boxes from the existing tracklets. In case some detection is associated with a target, the detected bounding box is used to update the target state via the Kalman filter. As in frame 2, the tracklets trk_1 and trk_2 are set to the two new bounding boxex b_1 and b_2 which are associated with predicted boxes b_3 and b_4 respectively. Otherwise, the target state is simply predicted without correction using the linear velocity model. For example, at time point 3, the detector misses detecting the white car due to an occlusion, however, the tracklet 2 is till assigned to box b_7 which contains part of the white car.



```
<<:det1 :det :b1>> a :car; :score 0.8;  sosa:resultTime 2.
<<:det3 :det :b3>> a :car; :score 0.7;  sosa:resultTime 2.
        <<:trk1 :trk :b2>>   sosa:resultTime 2.
        <<:trk2 :trk :b4>>   sosa:resultTime 2.
```

```
<<:det5 :det :b5>> a :car; :score 0.8; sosa:resultTime 3.
    <<:trk1 :trk :b6>> sosa:resultTime 3.
    <<:trk2  :trk :b7>> sosa:resultTime 3.
```

```
<<:det8   :det :b8 >> a :car; :score 0.8;  sosa:resultTime 4.
<<:det11 :det :b11>> a :car; :score 0.5;  sosa:resultTime 4.
<<:trk1 :trk :b9  >>  sosa:resultTime 4.
<<:trk2 :trk :b10>>  sosa:resultTime 4.
<<:trk3 :trk :b11>>  sosa:resultTime 4.
```

Fig. 2 A Semantic Visual Stream Snapshot. The red boxes are detected bounding boxes and the yellow boxes are tracked bounding boxes.

To emulate these MOT algorithms, CQELS 2.0 represents these association hypotheses by hard rules and soft rules that are translated into an optimization problem solved by an ASP Solver. For data abstraction, we use the neural-symbolic stream modelling practice in [15], which extends the standardised Semantic Sensor Network

Ontology (SSN) [11] to model video streams and intermediate processing states.The extension includes various vocabularies to specify the semantics of camera sensors, video frames, and tensors as shown in Figure 3.

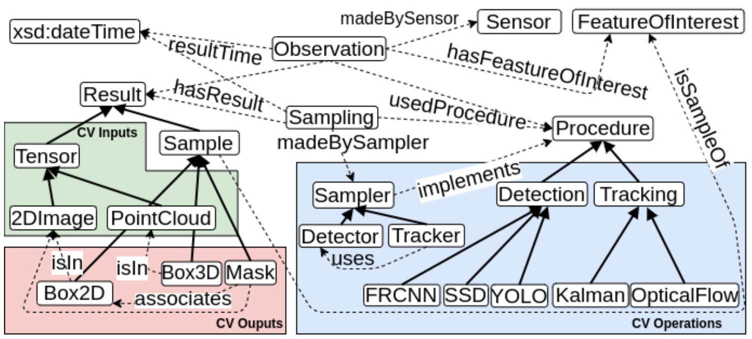


Fig. 3 Data abstraction of Neural-Symbolic stream with Semantic Sensor Network Ontology

A video stream fusion data pipeline can consume the data that is observed by a *Sensor* (i.e. a camera) as *a stream of observations* (represented as an *Observation*, e.g. a video frame). These video frames are represented as instances of the subclass $2DImage$ of the class $Tensor$ that inherits from the generic $Result$ class of SSN. These observations can then be fed into a probabilistic inference process such as a DNN model or a CV algorithm (represented as a *Procedure*) to provide derived stream elements which then represent $Sampling$ instances. For example, after being processed by the Feature Extractor (see Figure 1), the symbolically represented in RDF* format[3] of the frame 2 in the stream snapshot in Figure 2 is illustrated in Listing 1. Line 5 denotes that the detection model generates an output consisting of a bounding box b_1, object type *car* and confidence score 0.8. Line 10 presents that the box b_2 is predicted by a Kalman filter and is tracked by tracklet 1.

```
1  // time point/frame 2
2  :obs2 sosa:madeBySensor :cam;  sosa:resultTime 2.
3  :image2 a :2DImage; :image2 sosa:isResultOf :obs2.
4  <<:det1 :det :b1>> a :car; :score 0.8; sosa:resultTime 2;
5      sosa:usedProcedure :YOLO.
6  <<:det3 :det :b3>> a :car; :score 0.7; sosa:resultTime 2;
7      sosa:usedProcedure :YOLO.
8  <<:trk1 :trk :b2>> sosa:usedProcedure :Kalman; sosa:resultTime 2.
9  <<:trk2 :trk :b4>> sosa:usedProcedure :Kalman; sosa:resultTime 2.
```

Listing 1 Semantic Stream Serialization with RDF*

To formalise the reasoning process with a semantic representation of stream data, we need a temporal model that allows us to reason about the properties and features of objects. This model must account for the laws of the physical world and

[3] https://w3c.github.io/rdf-star/cg-spec/editors_draft.html

commonsense, which allow the system to handle incomplete information (e.g., if we do not see objects appearing in observations, or camera reads are missing).

Using the above RDF representations, we allow RDF&SPARQL developers to write soft rules with SHACL rule language[4] along with the extension of CQEL-QL. The extension we made here is replacing SPARQL CONSTRUCT with the corresponding CONSTRUCT of CQELS-QL which extends SPARQL with the window operators over RDF Stream. Moreover, extending SHACL for expressing ASP-like rules is aligned with the recent proposal for assigning SHACL to negation stable semantics [1]. Note that we extend CQESL-QL syntax in [14] with the keyword "NAF" to express the default negation of ASP. For example, the soft rule 1 in Listing 2 is used to trigger the event a car enters the "Field of View" of a camera and the car starts being tracked.

```
ssr:rule_w_1 a sh:NodeShape;
sh:rule [
  a sh:CQELSRule ;
  sh:prefixes ssr: ;
  sh:construct """
    CONSTRUCT {<<?O :enters <ssr:FoV>> @ ?T.}
    WHERE {
      STREAM <:ssr> {
        <<?Dt :det ?B >> @ ?T; :score ?S.
        ?B sosa:isSampleOf ?O ; a :car.
        FILTER (?S > 0.8)
      }
      NAF STREAM <:ssr> window[5 sec] {
          ?O :inFOV ssr:FoV.
      }
    }
    """ ;
] ;
```

Listing 2 Soft rule 1 - detect vehicles entering Field of View

To associate a detected bounding box B with an object O, we use soft rules that assert the triple «B sosa:isSampleOf O» based on *explained spatial, temporal, and visual appearance evidences*. Such rules can be used to represent *hypotheses* on temporal relations among detected objects in video frames following a tracking trajectory. When the object's movement is consistent with the constant velocity model, e.g., the Kalman filter used in SORT [3], and there is a detection associated with its trajectory, the fact «B sosa:isSampleOf O» is generated by rule 3. Here, $iou(B_1, B_2)$ states the IOU (intersection over union) condition of the bounding boxes B_1 and B_2 satisfy.

Furthermore, we can also emulate DeepSORT [25] via the soft rule 4 that can search for supporting evidences to link a newly detected bounding box from an occluded tracklet using visual appearance associations, e.g. frames 2 and 4 of Figure 2. For this, we search for pairs of bounding boxes from recently occluded tracklets w.r.t. visual appearance. As the search space of possible matches is large, we limit it by filtering the candidates based on their temporal and spatial properties. To this end,

[4] SHACL Advanced Features

we use rules with windows to reason about disconnected tracklets that have bounding boxes visually matched within a window of time points that are aligned with Deep-SORT's gallery of associated appearance descriptors for each tracklet. Based on this gallery of previously tracked boxes, the appearance-based discriminative metrics are computed to recover the identities after long-term occlusion, where the motion is less discriminative. Hence, to connect a newly detected bounding box B_1 that has a visual appearance match with another bounding box B_2 (represented by the fact «B_1 :vMatch B_2.» in line 13) of a discontinued tracklet T_2 (represented by «Trk_2 :ends T_e» in line 15) that ended 3 time points before.

```
ssr:rule_w_2 a sh:NodeShape ;
sh:rule [
  a sh:CQELSRule ;
  sh:prefixes ssr: ;
  sh:construct """
  CONSTRUCT { ?B1 sosa:isSampleOf ?O. }
  WHERE{
    STREAM <:ssr>{
      <<?Dt :det ?B2 >> @ ?T; :score ?S.
      <<?Trk :trk ?B1 >> @ ?T.
      ?Trk :trklet ?O.
      FILTER (?S>0.8 && iou (?B1,?B2))
    }
  }
  """ ;
] ;
```

Listing 3 Soft rule 2 - emulation of SORT algorithm [3].

```
1  ssr:rule_w_3 a sh:NodeShape ;
2  sh:rule [
3    a sh:CQELSRule ;
4    sh:prefixes ssr: ;
5    sh:construct """
6    CONSTRUCT { ?B1 sosa:isSampleOf ?O. }
7    WHERE{
8      STREAM <:ssr> @?Te window[5 sec] {
9        ?Trk2 :trk ?B2
10     }
11     STREAM <:ssr>{
12       <<?Trk1 :trk ?B1 >> @ ?T.
13       ?B1 :vMatch ?B2.
14       ?B2 sosa:isSampleOf ?O.
15       ?Trk2 :ends ?Te.
16       FILTER {?T<?Te+3}
17     }
18   }
19   """ ;
20 ] ;
```

Listing 4 Soft rule 3 - emulation of DEEPSORT algorithm [25].

4 System Architecture Overview

In this section, we present a specific architecture of CQELS 2.0 for the conceptual design in Figure 1. CQELS famework provides a platform-independent infrastructure to implement RDF-Stream Processing (RSP) engines for computing continuous queries expressed in CQELS-QL. The first version of CQELS accepts RDF streams as input and returns RDF streams or relational streams in the SPARQL format as output. CQELS 2.0 allows creating RDF streams by annotating extracted features from other data streams. The output RDF streams can be fed into any RSP engine, and the relational stream can be used by other relational stream processing systems.

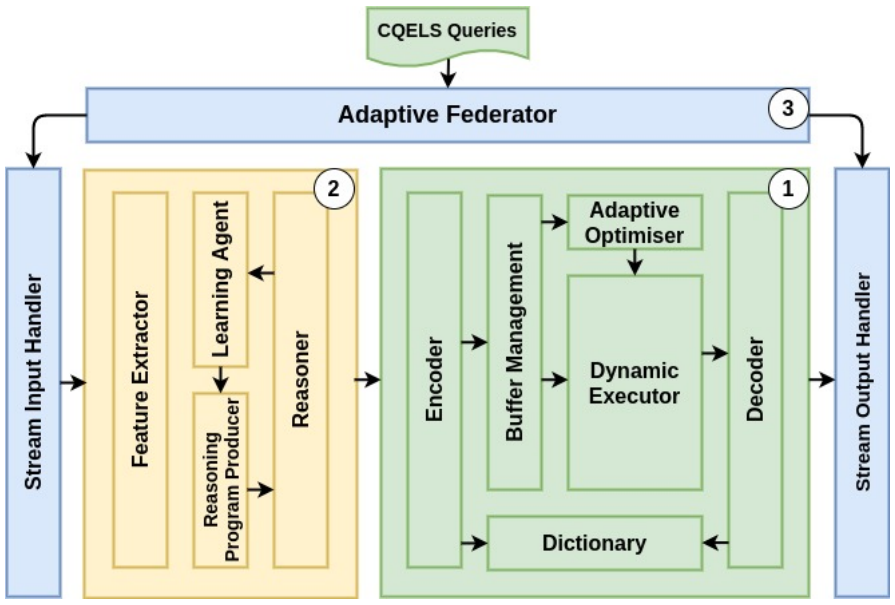


Fig. 4 System architecture of CQELS 2.0.

Figure 4 illustrates the overview architecture of CQELS 2.0. In general, CQELS 2.0 consists of three subsystems. Subsystem (1), the RDF stream processing processor, extends the stream processing primitives of its previous version [14] to accelerate the grounding phase of stream reasoning. For example, multiway joins are used to accelerate the incremental ground techniques [12]. The second subsystem (2) is the semantic stream reasoning component as presented in Section 2. Finally, subsystem (3) is a stream adaptive federator, which is described as follows.

Thanks to the platform-agnostic design of its execution framework [13], the core components are abstract enough to be seamlessly integrated with different RDF libraries in order to port the resulting system to different hardware platforms. For scalability, CQELS employs Storm[5] and HBase[6] as underlying software stacks for coordinating parallel execution processes to build an RSP engine on the cloud computing infrastructure, called CQELS Cloud [16]. To tailor the RDF-based data processing operations on edge devices (e.g, ARM CPU, Flash-storage), CQLES can be integrated in RDF4Led [17], a RISC style RDF engine for lightweight edge devices, to build Fed4Edge [20]. The whole Fed4Edge is smaller than 10MB and needs only 4–6 MB of RAM to process millions of triples on various small devices such as BeagleBone,[7] Raspberry PI.[8] Therefore, CQELS 2.0 includes an adaptive federation mechanism to enable the coordination of different hardware resources to process query processing pipelines by cooperatively delegating partial workloads to their peer agents.

The Adaptive Federator acts as the query rewriter, which adaptively divides the input query into sub queries. The rewriter then pushes down the operators as close to the streaming nodes as possible by following the predicate pushdown practice in common logical optimisation algorithms. The metadata subscribed by the other CQELS instances is stored locally. Similar to [7], such metadata allows the endpoint services of a CQELS engine to be discovered via the Adaptive Federator. When the Adaptive Federator sends out a subquery, it notifies the Stream Input Handler to subscribe and listens to the results returning from the subquery. On the other hand, the Stream Output Handler sends out the subqueries to other nodes or sends back the results to the requester.

Similar to cloud integration with CQELS Cloud [16], this federation design also covers elastic-scale delopment by using the new development of Apache Flink under BIFOLD project. In particular, we use the EMMA compiling and parallelizing for data flow systems in [10] to scale and optimize our processing pipelines in the cloud infrastructure. This will lay the foundation for integration the adaptive optimizer with the cloud-based stream scheduler and operation allocations.

With the support of the above subscription and discovery operations, a stream processing pipeline written in CQELS-QL can be deployed across several sites distributed in different locations: e.g., weather stations provide environmental sensory streams in various locations on earth. Each autonomous CQELS node gives access to data streams fed from the streaming nodes connecting to it. Such stream nodes can ingest a range of sensors, such as air temperature, humidity and carbon monoxide. When the stream data arrives, this CQELS node can partially process the data at its processing site and then forward the results as mapping or RDF stream elements to its parent node.

[5] Storm. https://storm.apache.org/

[6] Hbase. http://hbase.apache.org/.

[7] https://beagleboard.org/bone

[8] https://www.raspberrypi.org/

References

1. Andresel, M., Corman, J., Ortiz, M., Reutter, J.L., Savkovic, O., Simkus, M.: Stable model semantics for recursive SHACL. In: Y. Huang, I. King, T. Liu, M. van Steen (eds.) WWW ’20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020, pp. 1570–1580. ACM / IW3C2 (2020)
2. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. EURASIP Journal on Image and Video Processing **2008**, 1–10 (2008)
3. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: ICIP, pp. 3464–3468 (2016)
4. Bishop, G., Welch, G., et al.: An introduction to the kalman filter. Proc of SIGGRAPH, Course **8**(27599-23175), 41 (2001)
5. Ciaparrone, G., Sánchez, F.L., Tabik, S., Troiano, L., Tagliaferri, R., Herrera, F.: Deep learning in video multi-object tracking: A survey. Neurocomputing (2019)
6. Dao-Tran, M., Le Phuoc, D.: Towards enriching cqels with complex event processing and path navigation. In: HiDeSt@ KI, pp. 2–14. Citeseer (2015)
7. Dell’Aglio, D., Le Phuoc, D., Le-Tuan, A., Intizar Ali, M., Calbimonte, J.P.: On a web of data streams (2017)
8. Dell’Aglio, D., Dao-Tran, M., Calbimonte, J.P., Le Phuoc, D., Della Valle, E.: A query model to capture event pattern matching in rdf stream processing query languages. In: European Knowledge Acquisition Workshop, pp. 145–162. Springer (2016)
9. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y.: Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet Things J. **7**(8), 7457–7469 (2020). DOI 10.1109/JIOT.2020.2984887. URL https://doi.org/10.1109/JIOT.2020.2984887
10. Gévay, G.E., Rabl, T., Breß, S., Madai-Tahy, L., Quiané-Ruiz, J., Markl, V.: Efficient control flow in dataflow systems: When ease-of-use meets high performance. In: 37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021, pp. 1428–1439. IEEE (2021). DOI 10.1109/ICDE51399.2021.00127. URL https://doi.org/10.1109/ICDE51399.2021.00127
11. Haller, A., Janowicz, K., Cox, S.J.D., Lefrançois, M., Taylor, K., Phuoc, D.L., Lieberman, J., García-Castro, R., Atkinson, R., Stadler, C.: The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semantic Web **10**(1), 9–32 (2019). DOI 10.3233/SW-180320. URL https://doi.org/10.3233/SW-180320
12. Le-Phuoc, D.: Adaptive optimisation for continuous multi-way joins over rdf streams. Companion Proceedings of the The Web Conference 2018 (2018)
13. Le-Phuoc, D., Dao-Tran, M., Le Van, C., Le Tuan, A., Manh Nguyen Duc, T.T.N., Hauswirth, M.: Platform-agnostic execution framework towards rdf stream processing. In: RDF Stream Processing Workshop at ESWC2015 (2015)
14. Le-Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. ISWC’11 (2011)
15. Le-Phuoc, D., Eiter, T., Le-Tuan, A.: A scalable reasoning and learning approach for neural-symbolic stream fusion. Proceedings of the AAAI Conference on Artificial Intelligence **35**(6), 4996–5005 (2021). URL https://ojs.aaai.org/index.php/AAAI/article/view/16633
16. Le-Phuoc, D., Quoc, H.N.M., Le Van, C., Hauswirth, M.: Elastic and scalable processing of linked stream data in the cloud. In: International Semantic Web Conference, pp. 280–297. Springer (2013)
17. Le-Tuan, A., Hayes, C., Wylot, M., Le-Phuoc, D.: Rdf4led: An rdf engine for lightweight edge devices. In: IOT ’18 (2018)
18. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. In: ECCV, pp. 21–37 (2016)
19. News, B.: CCTV: Too many cameras useless, warns surveillance watchdog Tony Porter. https://www.bbc.com/news/uk-30978995 (2015). Accessed: 2021-07-25
20. Nguyen-Duc, M., Le-Tuan, A., Calbimonte, J.P., Hauswirth, M., Le-Phuoc, D.: Autonomous rdf stream processing for iot edge devices. In: Semantic Technology, pp. 304–319. Springer, Cham (2020)

21. Nguyen-Duc, M., Le-Tuan, A., Hauswirth, M., Le-Phuoc, D.: Towards autonomous semantic stream fusion for distributed video streams. In: Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems, pp. 172–175 (2021)
22. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. CoRR (2015)
23. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: CVPR, pp. 6517–6525 (2017)
24. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR (2015)
25. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: ICIP, pp. 3645–3649 (2017)
26. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proc. IEEE **107**(8), 1738–1762 (2019). DOI 10.1109/JPROC.2019.2918951. URL https://doi.org/10.1109/JPROC.2019.2918951