

Urban Drone Traffic Safety with Semantic Web Enhanced Stream Reasoning through DatalogMTL^{RDF}

Volodymyr Kadzhaia, Pieter Bonte

Departement of Computer Science, KU Leuven Kulak, Belgium

Abstract

Drone traffic management is an emerging safety-critical domain in which autonomous and remotely piloted aircraft must be continuously monitored for geofence violations, control link failures, collision risks, and deviations from filed flight plans. Detecting these situations requires reasoning over multiple heterogeneous data streams in real time, correlating temporal patterns across irregular sensor signals, and integrating background knowledge about airspace regulations and infrastructure. Stream reasoning offers a principled approach to this challenge, and DatalogMTL stands out as a particularly promising formalism, offering expressive metric temporal operators that can capture complex time-sensitive safety rules. However, DatalogMTL in its current form is ill-suited to practical deployment in real-world streaming scenarios.

In this demo paper we present DatalogMTL^{RDF}, a stream-native extension of DatalogMTL for RDF event streams, and demonstrate its application to drone traffic safety monitoring. DatalogMTL^{RDF} bridges the gap between the theoretical expressivity of DatalogMTL and the practical demands of real sensor streams, enabling rules that correctly detect sustained geofence violations, control link loss, and multi-drone collision risks.

Keywords

Stream Reasoning, Drones, DatalogMTL, RDF Stream Processing

1. Introduction

Autonomous and remotely piloted aircraft are becoming a routine part of urban airspace, from delivery drones and infrastructure inspection platforms to emergency response and surveillance systems. As drone density grows, regulatory bodies across Europe and beyond are establishing Unmanned Traffic Management (UTM) frameworks that require continuous, automated supervision of every flight in shared airspace. The supervision task is fundamentally temporal: safety violations rarely manifest as single isolated events, but as patterns unfolding over time. A drone briefly entering a restricted zone may be a navigation glitch, while one lingering there for thirty seconds is an incident. A momentarily lost telemetry signal is unremarkable, while a silence lasting ten seconds over a hospital is an emergency. Detecting such patterns requires more than continuous querying: it requires reasoning over heterogeneous data streams in real time, correlating temporal patterns across irregular sensor signals, and integrating background knowledge about airspace regulations and infrastructure. Stream reasoning addresses this need by combining the expressive power of knowledge representation formalisms with the continuous, incremental evaluation required by data streams [1]. Among the available temporal reasoning formalisms, DatalogMTL stands out as a particularly promising candidate [2]: it extends Datalog with metric temporal operators, is decidable, and its forward-propagating fragment is tractable in data complexity. However, DatalogMTL in its current form is ill-suited to practical deployment in real-world streaming scenarios, limiting its applicability precisely where it is needed most. This paper addresses this gap through DatalogMTL^{RDF}, a stream-native extension of DatalogMTL for RDF event streams, and demonstrates three concrete benefits it brings to drone traffic safety monitoring.

Temporal pattern expressivity beyond windows. Traditional RDF Stream Processing (RSP) systems [3] reason over fixed-size time windows, which cannot express patterns such as *a drone has been off its flight plan ever since it crossed into a restricted zone, or at some point in the last hour there was*

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, Sep 2026, Ghent, Belgium

✉ volodymyr.kadzhaia@kuleuven.be (V. Kadzhaia); pieter.bonte@kuleuven.be (P. Bonte)

id 0009-0000-1715-7782 (V. Kadzhaia); 0000-0002-8931-8343 (P. Bonte)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

a sustained five-minute geofence violation. DatalogMTL^{RDF} expresses these naturally through metric temporal operators, which operate over arbitrary time intervals without requiring fixed window sizes.

Background knowledge integration. Drone safety rules are not purely about position streams. They require reasoning over background knowledge: which zones are restricted, what class of drone is permitted in which airspace, and what weather conditions apply. Knowledge graphs provide a natural representation layer for this. While RSP-QL [4] allows windows to be joined with static RDF graphs, and DatalogMTL admits rules over non-temporal predicates, neither cleanly addresses where stream-derived facts end and static knowledge begins. DatalogMTL^{RDF} makes this distinction explicit: we introduce the idea of *stream shapes* which declare which facts originate from named streams, while remaining facts are treated as background knowledge. Rules can then join across the two sources within a single rule body, with provenance preserved.

Gap detection and staleness reasoning. A drone losing its telemetry signal is not merely a missing data point: it is a safety-critical event. Sensors transmit at variable frequencies and may go silent deliberately or due to failure. Under standard point semantics, absent data is simply invisible to the reasoner. DatalogMTL^{RDF} introduces *step function semantics*, grounded in temporal database theory [5], in which a reading persists until superseded or until a configurable staleness threshold is exceeded. This enables rules such as *alert if no telemetry has been received for ten seconds while the drone was last seen near a restricted zone*, which are inexpressible under point semantics.

Contributions. We introduce DatalogMTL^{RDF} , a stream-native extension of DatalogMTL that makes the formalism practically applicable to real-world RDF event streams, and implement it in **Kolibrie** [6], an open-source RDF stream reasoning engine. We demonstrate the system on a simulated urban drone traffic safety scenario, showing temporal safety rules firing in real time over a live event stream involving geofence violations, control link loss, and multi-drone collision risks.

2. Background on DatalogMTL

DatalogMTL [2] extends Datalog with operators from Metric Temporal Logic, enabling rules that reason about the temporal evolution of facts. A DatalogMTL program consists of a set of rules of the form

$$A_0 \leftarrow A_1, \dots, A_n$$

where the head A_0 is an atom and the body atoms $A_1, \dots, A_n$ may be wrapped in metric temporal operators. Each atom $P(c_1, \dots, c_k)$ holds at one or more timestamps drawn from the rational timeline. Facts in the dataset are timestamped atoms; derived facts inherit their timestamp from the rule evaluation.

DatalogMTL^{RDF} uses the past-only fragment of DatalogMTL , which is appropriate for stream reasoning since only past events are available. We focus on three operators:

- $\boxplus_{[a,b]} A$: A holds at *every* timepoint in the interval $[t - b, t - a]$. Used for sustained patterns such as “continuously occupied for the last hour.”
- $\boxtimes_{[a,b]} A$: A holds at *some* timepoint in $[t - b, t - a]$. Used for existential patterns such as “temperature exceeded the threshold at least once in the last fifteen minutes.”
- $A_1 \mathcal{S}_{[a,b]} A_2$: A_2 holds at some $t' \in [t - b, t - a]$, and A_1 holds at every $t'' \in (t', t]$. Used for patterns such as “has been off its flight plan ever *since* entering the restricted zone.”

3. DatalogMTL^{RDF}

The DatalogMTL^{RDF} data model rests on three core concepts: *named streams*, *stream shapes*, and the *induced dataset* that connects them back to standard DatalogMTL .

Named streams. A named stream is an unbounded sequence of *RDF event graphs* arriving at successive timestamps, identified by an IRI. Each event is a small RDF graph rather than a single triple, reflecting the way real observations are composed.

Stream shapes. A stream shape declares how to interpret events on a named stream: which co-arriving triples constitute a single logical event, and when a new event represents an update of a previous observation rather than an independent one. It consists of three elements:

- an *event pattern*, a basic graph pattern describing the expected structure of events;
- a *channel key*, an ordered tuple of variables from the event pattern that identifies one logical channel within the stream;
- a *staleness threshold*, the duration after which a reading is considered expired if no new event has arrived.
- a *default event*, an RDF graph template that becomes valid when the staleness threshold elapses, with channel key variables bound from the expired channel.

The channel key is the temporal analog of a *primary key*: it identifies which events represent successive observations of the same logical entity. In the drone scenario, the channel key (`?drone`) identifies one telemetry channel per drone, ensuring that observations from different drones are tracked independently.

The default event handles what should hold when a channel falls silent. This mechanism is the stream reasoning analogue of the Last Will and Testament pattern in MQTT¹, in which a client declares in advance a payload to be published when its connection is unexpectedly terminated. Stream shapes generalize this to RDF and to the temporal setting by declaring a default graph that becomes valid when the channel expires and remains valid until a new event arrives.

Induced dataset and step function semantics. Given a stream and a shape, DatalogMTL^{RDF} derives an *induced dataset*: a set of triples annotated with validity intervals over which DatalogMTL 's temporal operators are naturally evaluated. The shape itself dictates how the intervals are constructed, following *step function semantics* [5]: when an event arrives on a channel, its triples become valid from that timestamp onward, and remain valid until either a subsequent event on the same channel supersedes them, the staleness threshold τ elapses without a new event, or the current observation time is reached. Adjacent intervals with identical content are coalesced into a single maximal interval.

This semantics has two important consequences. A reading *persists between observations*, so temporal operators correctly evaluate over the full duration of a reading rather than requiring a fact to be re-asserted at every time-stamp. At the same time, a gap in the stream becomes *semantically meaningful*: if no event arrives within the staleness threshold, the channel expires, enabling rules to detect the absence of expected observations. The staleness threshold τ is thus a property of the stream itself, encoded once in the shape, rather than a parameter chosen per query.

4. The Drone Safety Demonstrator

The demonstrator² instantiates DatalogMTL^{RDF} in a simulated urban airspace monitoring scenario, running on the Kolibrie engine [6]. It is designed to showcase the three benefits identified in the introduction: temporal pattern expressivity, background knowledge integration, and gap detection.

4.1. Scenario

The scenario simulates a map of the city of Ghent with two active drones operating under Unmanned Traffic Management (UTM) supervision. The airspace contains three marked restricted zones: a hospital, a government building, and a temporary no-fly zone over a public event. Each drone has a filed flight plan registered in a background knowledge graph alongside vehicle metadata such as drone class, operator identity, and authorised altitude bounds. We model drones, telemetry, flight plans, and alarms using Dronetology [7], observations using SOSA/SSN and supplement these with a small auxiliary vocabulary (prefix `utm:`) for telemetry attributes not directly modelled by Dronetology. Telemetry

¹<https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/>

²A video walkthrough of the live demo is available at https://drive.google.com/file/d/11r7jgzHfzzLdrj9oMJ2g_98Qbm6-ybbE.

messages arrive on a named stream at a nominal rate of one message per second per drone, each modelled as a small RDF graph linking the drone to its position, altitude, battery level, and AIS status.

4.2. Stream Shape

A single stream shape declares how telemetry events are interpreted:

```
StreamShape :TelemetryStream
  StreamIRI:      <http://utm.example.org/telemetry>
  EventPattern:
    ?obs a sosa:Observation ;
        sosa:madeBySensor ?drone ;
        sosa:hasResult ?tlm .
    ?drone a dront:Drone .
    ?tlm a dront:Telemetry ;
        utm:position ?pos ;
        utm:altitude ?alt ;
        utm:aisStatus ?status .
  ChannelKey:    (?drone)
  MaxGap:        10000ms
  DefaultEvent:
    ?drone utm:channelStatus utm:expired .
```

The channel key (`?drone`) ensures that telemetry from each drone forms an independent channel. The staleness threshold (`MaxGap`) ensures that when the threshold elapses without a new event, the default event activates: the triple `?drone utm:channelStatus utm:expired` becomes valid.

4.3. Safety Rules

Three rules illustrate the three benefits in turn. We write rules in DatalogMTL^{RDF} syntax, where atoms are RDF triple patterns.

Sustained geofence violation (background knowledge). The first rule joins live telemetry against the background knowledge graph, which marks specific zones as restricted:

$$(?d, :violatedZone, ?z) \leftarrow \exists_{[0,30000]} (?d, dront:inZone, ?z), (?z, dront:status, dront:restricted)$$

A violation is derived when a drone has remained in a zone continuously for thirty seconds, *and* the zone is classified as restricted in the knowledge graph. The temporal operator handles the duration condition, while the non-temporal triple pattern matches the static classification stored in the KG. Both are evaluated within the same rule body, with provenance preserved.

Control link loss (gap detection). The second rule fires when a drone's channel has been in the expired state continuously for ten seconds:

$$(?d, :status, :linkLost) \leftarrow \exists_{[0,10000]} (?d, dront:channelStatus, dront:expired)$$

The triple matched by the body is produced by the default event mechanism declared in the stream shape: when no telemetry arrives within the staleness threshold, the channel transitions to the expired state and the corresponding triple becomes valid in the induced dataset. The Box operator confirms that this state has persisted for ten seconds, distinguishing a sustained link loss from a momentary jitter. The rule is positive and matches a real triple in the induced dataset, without requiring negation.

Off-course since restricted entry (temporal expressivity). The third rule uses the Since operator to capture a pattern that fixed-window stream processors cannot express:

$$(?d, :status, :offCourse) \leftarrow (?d, dront:onFlightPlan, :false), \\ S_{[0,600000]}(?d, dront:enteredZone, ?z), (?z, dront:status, dront:restricted)$$

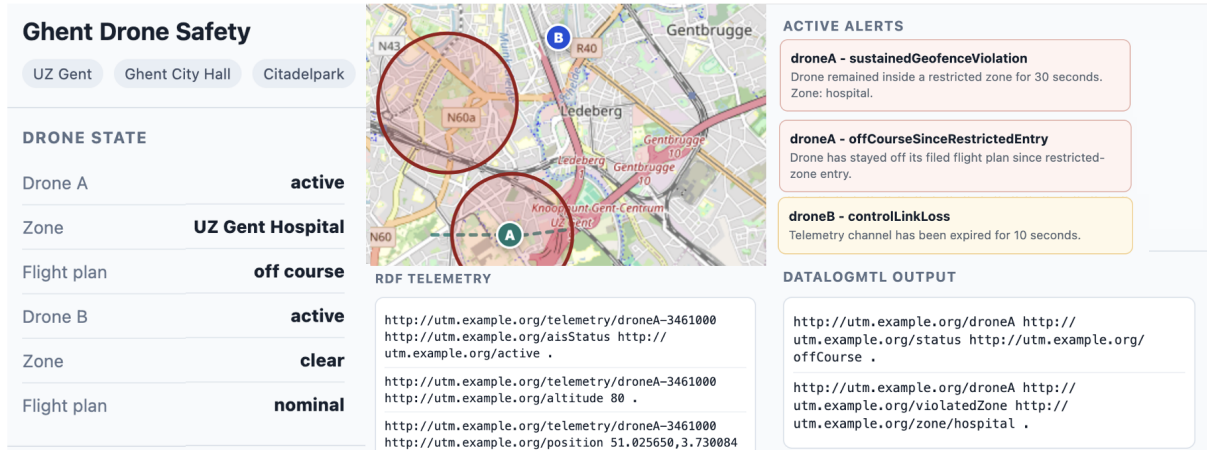


Figure 1: The DatalogMTL^{RDF} drone safety demonstrator. Drone A has triggered a sustained geofence violation over the hospital zone; Drone B has lost its telemetry link.

The rule derives that drone $?d$ is off course if it entered a restricted zone within the last ten minutes *and* has remained off its filed flight plan ever since. The interval between the triggering event and the present is determined by the data, not by a pre-set window size.

4.4. The Interface

Figure 1 shows the demonstrator interface during a scripted scenario. The map panel displays the urban area with drones, restricted zones, and filed flight plans. The event log shows live RDF telemetry events arriving on the stream. The alerts panel surfaces derived facts as rules fire, annotated with the timestamp and the rule responsible.

The live demo presents a scripted scenario in which Drone A follows a fixed flight plan, while the user can interactively drag and drop Drone B to arbitrary locations on the map. This allows attendees to trigger and observe the three classes of safety events in real time: a drone deviating from its filed flight plan, a drone entering a restricted zone, and a drone losing its telemetry signal. As conditions change, alerts appear and clear in the alerts panel, demonstrating that the temporal operators are continuously re-evaluated over the live stream.

Declaration on Generative AI: The authors used Claude for paraphrasing and aiding with the visualization of the demo, then reviewed and edited the work and take full responsibility for its content.

References

- [1] P. Bonte, et al., Grounding stream reasoning research, Transactions on Graph Data and Knowledge (TGDK) 2 (2024) 1–47.
- [2] S. Brandt, et al., Ontology-based data access with a horn fragment of metric temporal logic, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.
- [3] P. Bonte, C. Callé, O. Curé, H. Kondylakis, R. Tommasini, Languages and systems for rdf stream processing, a survey: P. bonte et al., The VLDB Journal 34 (2025) 50.
- [4] D. Dell’Aglio, et al., RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems, Int. J. Semantic Web Inf. Syst. 10 (2014).
- [5] C. S. Jensen, R. T. Snodgrass, Semantics of time-varying information, Information Systems (1996).
- [6] V. Kadzhaia, P. Bonte, Neuro-symbolic stream reasoning with kolibri, in: European Semantic Web Conference, Springer, 2026, pp. 39–57.
- [7] D. Martín-Lammerding, J. J. Astrain, A. Córdoba, Dronetology: A domain ontology for uas applications, Knowledge-Based Systems (2025) 114074.