# Traffic Management using RTEC in OWL 2 RL


Bernard Gorman
IBM Research, Dublin
berngorm@ie.ibm.com

Jakub Marecek
IBM Research, Dublin
jakub.marecek@ie.ibm.com

Jia Yuan Yu
IBM Research, Dublin
jy@osore.ca


*Introduction.* In a number of domains, including traffic management, event processing and situational reporting are particularly demanding. This is due to the volumes and realiability of streamed spatio-temporal data involved, ranging from sensor readings, news-wire reports, police reports, to social media, as well as the complexity of the reasoning required. Human, rather than artificial, intelligence is hence still used to an overwhelming extent.

A number of specialised event-processing languages and reasoners have been proposed, extending RDF and SPARQL. These include SPARQL-ST [11], Temporal RDF [14] and T-SPARQL [7], Spatio-temporal RDF and stSPARQL [9]. For even more elaborate extensions, see e.g. [12, 2, 10]. Often, these extensions rely on custom parsers for the languages and on custom Prolog-based implementations of reasoners. Yet, none of these extensions has gained a wide adoption.

We argue that such specific languages and reasoners go against the principle of a general-purpose description logics and general-purpose reasoners [3]. We propose a rewriting of RTEC, the event processing calculus [2], from Prolog to OWL 2 RL [8], which is the only profile of the Web Ontology Language, for which there exist very efficient reasoners.

*RTEC.* Artikis et al. [2] proposed Event Calculus for Run-Time reasoning (RTEC) as a calculus for event processing. Prolog-based implementations, where event processing is triggered asynchronously and the derived events are produced in a streaming fashion, are readily available [1]. In order to make this paper self-contained, we summarise its principles beyond the very basics [6].

Time is assumed to be discretised and space is represented by GPS coordinates. All predicates in RTEC are defined by Horn clauses [6], which are the implications of a head from a body, $h_1, \ldots, h_n \leftarrow b_1, \ldots, b_m$, where $0 \leq n \leq 1$ and $m \geq 0$. All facts are predicates with $m = 0$ and $n = 1$, such as `move(B1, L1, O7, 400)`, which means that a particular bus B1 is running on a particular line L1 with a delay of 400 seconds, as operated by operator O7. Similarly, `gps(B1, 53.31, -6.23, 0, 1)` means that the bus B1 is at the given, its direction is forwards (0) and there is congestion (1). Based on such facts, one formulates rules, i.e. Horn clauses with $m > 0$ and $n = 1$, for the processing of instantaneous events or non-instantaneous fluents.The occurrence of an event $E$, which is an inferred Horn clause with $m > 0$ and $n = 1$, at a fixed time $T$, is given by rules using `happensAt(E, T)`. The occurrence of a fluent $F$ is at a finite list $I$ of intervals, is given using `holdsFor(F=V, I)`. Simple fluents, which hold in a single interval, are given by `initiatedAt(E, T)` and `terminatedAt(E, T)`. For an overview of the predicates, please see Table 1.

Notice that Horn clauses can be used to define complex events, such as the sharp increase in the delay of a bus parametrised by thresholds `t, d` for time and delay:

```
happensAt(delayIncrease(Bus, X, Y, Lon, Lat), T)
:- happensAt(move(Bus, _, _, Delay0), T0),
   holdsAt(gps(Bus, X, Y, _, _)=true, T0),
   happensAt(move(Bus, _, _, Delay), T),
   holdsAt(gps(Bus, Lon, Lat, _, _)=true, T),
   Delay - Delay0 > d,
   0 < T - T0 < t
```

where comma denotes conjunction, `_` is the anonymous variable, and `:-` denotes implication.

The complex events can be processed in a custom Prolog-based implementation [1], or as we show later, a OWL 2 RL reasoner [16]. In the Prolog-based implementation, one rewrites the inputs as facts, and leaves the reasoning about `delayIncrease` up to a Prolog interpreter. The resulting interactions between the ontology tools, Prolog interpreter, and rewriting among them are frail and challenging to debug, though.

*RTEC in OWL 2 RL.* It has long been known that Horn clauses can be rewritten into and queried in OWL 2. Recently, it has been shown [15] that Horn clauses can be rewritten in OWL 2 RL, a tractable profile of OWL. This rewriting allows for sound and complete reasoning, c.f. Theorem 1 of [16]. Moreover, the reasoning is very efficient, empirically.

The rewriting of Zhou et al. [16] proceeds via Datalog$^{\pm,\vee}$

**Table 1: Main predicates of RTEC. Cited loosely from [1].**

| Predicate | Meaning |
|---|---|
| `happensAt(E, T)` | Event $E$ occur s at time $T$ |
| `holdsAt(F=V, T)` | The value of fluent $F$ is $V$ at time $T$ |
| `holdsFor(F=V, I)` | The list $I$ of intervals for which $F = V$ holds |
| `initiatedAt(F=V, T)` | Fluent $F = V$ is initiated at $T$ |
| `terminatedAt(F=V, T)` | Fluent $F = V$ is terminated at $T$ |
| `relative_complement_all (I0, L, I)` | The list $I$ of intervals is obtained by complementing $i \in I0$ within ground set $L$ |
| `union_all(L, I )` | The list $I$ of intervals is the union of those in $L$ |
| `intersect_all(L, I )` | The list $I$ of intervals is the intersection of those in $L$ |

[4] and Datalog [6] proper into OWL 2 RL. Instead of goals in Prolog, which are Horn clauses with $m > 0$ and $n = 0$, one uses conjunctive queries in OWL 2 RL. Formally, Datalog$^{\pm,\vee}$ has first-order sentences of the form $\forall x \exists y$ s.t. $C_1 \wedge \cdots \wedge C_m \leftarrow B$, where $B$ is an atom with variables in $x$, which is neither $\perp$ nor an inequality. Conjunctive query (CQ) with distinguished predicate $Q(y)$ is $\exists y \phi(x, y)$ and $\phi(x, y)$ a conjunction of atoms without inequalities. In the example above, the Datalog$^{\pm,\vee}$ rule is:

∃ T', D, D' { ∃ a, b (`happensAt`(`move`(Bus, a, b, D'), T')) ∧
∃ c, d (`holdsAt`(`gps`(Bus, X, Y, c, d)=true, T')) ∧
∃ e, f (`happensAt`(`move`(Bus, e, f, D), T)) ∧
∃ g, h (`holdsAt`(`gps`(Bus, Lon, Lat, g, h)=true, T)) ∧
D - D' > `d` ∧
0 < T - T' < `t` }
← `happensAt`(`delayIncrease`(Bus, X, Y, Lon, Lat), T),

where all free variables (Bus, X, Y, Lon, Lat, T) are universally quantified. Following this line of work [15], we rewrite RTEC into OWL 2 RL.

This is the first ever translation of RTEC or any similar spatio-temporal event-processing logic to OWL 2 RL, as far as we know. In a companion paper co-authored with the staff at Dublin City Council [1], we describe an extensive traffic management system, where we employ RTEC in traffic management.

*Conclusions.* The value and scalability of spatio-temporal event processing over streaming data has been demonstrated a number of times [13, 5, 1]. Notice, however, that there remains a considerable gap between first prototypes specific to a particular city and a general-purpose methodology or tools. General-purpose reasoners using RTEC in OWL 2 RL may lack the performance of custom-tailored reasoners, capable of dealing with gigabytes of data at each time-step, but offer a handy tool for customising, prototyping, and debugging systems based on RTEC. The translation of Horn clauses to OWL 2 RL is clearly applicable to a number of other event-processing calculi based on Prolog [11, 14, 7, 9]. This approach may hence weill set the agenda in event processing more broadly.

# 1. REFERENCES


[1] A. Artikis et al. Heterogeneous stream processing and crowdsourcing for urban traffic management. In *EDBT*, pages 712–723, 2014.

[2] A. Artikis, M. Sergot, and G. Paliouras. Run-time composite event recognition. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 69–80. ACM, 2012.

[3] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In *Mechanizing Mathematical Reasoning*, pages 228–248. Springer, 2005.

[4] A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. *Logic in Computer Science, Symposium on*, 0:228–242, 2010.

[5] A. Del Bimbo, A. Ferracani, D. Pezzatini, F. D'Amato, and M. Sereni. Livecities: Revealing the pulse of cities by location-based social networks venues and users analysis.

[6] D. M. Gabbay, C. J. Hogger, and J. A. Robinson. *Handbook of Logic in Artificial Intelligence and Logic Programming: Volume 5: Logic Programming Volume 5: Logic Programming*. Oxford University Press, 1998.

[7] F. Grandi. T-sparql: A tsql2-like temporal query language for rdf. In *ADBIS (Local Proceedings)*, 2010.

[8] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. Owl 2: The next step for owl. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008.

[9] M. Koubarakis and K. Kyzirakos. Modeling and querying metadata in the semantic sensor web: The model strdf and the query language stsparql. In *The semantic web: research and applications*, pages 425–439. Springer, 2010.

[10] G. Meditskos, S. Dasiopoulou, V. Efstathiou, and I. Kompatsiaris. Ontology patterns for complex activity modelling. In *Theory, Practice, and Applications of Rules on the Web*, pages 144–157. Springer, 2013.

[11] M. Perry, P. Jain, and A. P. Sheth. Sparql-st: Extending sparql to support spatiotemporal queries. In *Geospatial semantics and the semantic web*, pages 61–86. Springer, 2011.

[12] M. Rinne. Sparql update for complex event processing. In *The Semantic Web–ISWC 2012*, pages 453–456. Springer, 2012.

[13] S. Tallevi-Diotallevi, S. Kotoulas, L. Foschini, F. Lécué, and A. Corradi. Real-time urban monitoring in dublin using semantic and stream technologies. In *The Semantic Web – ISWC 2013*, pages 178–194. Springer Berlin Heidelberg, 2013.

[14] J. Tappolet and A. Bernstein. Applied temporal rdf: Efficient temporal querying of rdf data with sparql. In *The Semantic Web: Research and Applications*, pages 308–322. Springer, 2009.

[15] Y. Zhou, B. Cuenca Grau, I. Horrocks, Z. Wu, and J. Banerjee. Making the most of your triple store: Query answering in owl 2 using an rl reasoner. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 1569–1580, 2013.

[16] Y. Zhou, Y. Nenov, B. C. Grau, and I. Horrocks. Complete query answering over horn ontologies using a triple store. In *International Semantic Web Conference (1)*, pages 720–736, 2013.