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 reliability 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 \( \text{move}(B1, L1, O7, 400) \), 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, \( \text{gps}(B1, 53.31, -6.23, 0, 1) \) means that the bus B1 is at the given, its direction is forwards (0) and there is congestion (1).

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:

\[
\text{happensAt}(\text{delayIncrease}(\text{Bus}, X, Y, \text{Lon}, \text{Lat}), T) :- \text{happensAt}(\text{move}(\text{Bus}, X, Y, \text{Delay}0), T0), \\
\text{happensAt}(\text{gps}(\text{Bus}, X, Y, \text{Lon}, \text{Lat})=\text{true}, T0), \\
\text{happensAt}(\text{move}(\text{Bus}, X, Y, \text{Delay}), T), \\
\text{happensAt}(\text{gps}(\text{Bus}, \text{Lon}, \text{Lat}, \text{Lon}, \text{Lat})=\text{true}, T), \\
\text{Delay} - \text{Delay}0 > d, \\
0 < T - T0 < t
\]

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 \( \text{delayIncrease} \) 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\(^{\pm,\lor} \).
∀x, y (ϕ(x, y) ∧ C) is the example above, the Datalog
positively 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
trafficking management system, where we employ RTEC in tra-
fic 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 well 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,
composite event recognition. In Proceedings of the 6th
ACM International Conference on Distributed Event-Based
ontology languages for the semantic web. In Mechanizing
[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.
M. Sereni. Livecities: Revealing the pulse of cities by
location-based social networks venues and users analysis.
of Logic in Artificial Intelligence and Logic Programming:
Volume 5: Logic Programming Volume 5: Logic
[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
metadata in the semantic sensor web: The model strdf and
the query language strsparql. In The semantic web: research
[10] G. Meditskos, S. Dasiopoulou, V. Efthathiou, and
I. Kompatsiaris. Ontology patterns for complex activity
modelling. In Theory, Practice, and Applications of Rules
sparql to support spatiotemporal queries. In Geospatial
semantics and the semantic web, pages 61–86. Springer,
2011.
2012.
and A. Corradi. Real-time urban monitoring in dublin
using semantic and stream technologies. In The Semantic
Heidelberg, 2013.
Efficient temporal querying of rdf data with sparql. In The
Semantic Web: Research and Applications, pages 308–322.
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,
query answering over horn ontologies using a triple store. In
International Semantic Web Conference (1), pages 720–736,
2013.