RoKEx: Robust Application-layer Knowledge Exchange

Peter McBrien, Nikos Rizopoulos, and Andrew Charles Smith
Dept. of Computing, Imperial College London,
Exhibition Road, London, SW7 2AZ

www.doc.ic.ac.uk

June 30, 2009

Abstract

Decisions in a dynamic environment may be based on information coming from a number of different knowledge sources described using different knowledge representation languages. This paper describes a common framework in which the data and rules, both static and dynamic, that may exist in disparate knowledge sources may be represented. We focus on two commonly used knowledge representation languages, namely SQL and OWL-DL. We have chosen these as examples because they make different assumptions about the knowledge they hold and we use them to show that our framework can represent knowledge under these different assumptions. **Keywords**: Knowledge Representation, OWL-DL, SQL

1 Introduction

There are many existing techniques to solve the problem of integrating *data* from heterogeneous sources. A popular approach, and one which we adopted to solve this problem in the RoDEx project[**?**], is to use some form of common data model as an intermediary [**?**]. In some cases we can add rules to our database that allow us to infer new facts from the existing data. We refer to this combination of data and rules as a **knowledge base**. This addition of rules makes the integration process much harder. In this paper we present a common data model in which to represent knowledge from different knowledge bases that will allow us to integrate the knowledge in the next phase of the project.

We start by presenting a motivating example. We then describe in detail how we differentiate between schemas which adopt the open world and closed world assumptions as well as those that use the unique name assumption and those that do not. We then go on to describe how we represent both static and dynamic rules from different knowledge bases in our system. We finish with some conclusions.

2 Motivating Example

There is an area where vehicles are monitored, where vehicles can be friendly military (FMV), friendly civilian (FCV) or hostile (HV) vehicles. There are two sources — one an array of ground sensors and the other a satellite tracking system — which provide data and knowledge regarding the area and its vehicles.

The ground sensors can detect vehicles and identify their markings, *i.e.* they provide data such as that shown in Table 1. The sensors provide: a reading id, a position, the marking for the vehicle, the direction and the velocity for the vehicle, and a timestamp. A knowledge source, illustrated in Figure 1, is also available to reason about the sensor data. The rules

sensor_data					
Id	X	Y	time	heading	speed
1	10	15	10:23	sw	40
2	10	18	10:25	w	50
3	10	15	9:00	n	90
4	10	20	9:14	s	20
5	10	15	8:02	s	35
6	10	15	8:40	nw	100

markings	
Id	Mark
1	+
2	<
3	+
5	+

Table 1: Sensor array data, containing the data feed in sensor_data, and the result of image processing for marking detection in marking

identify the vehicles: a + marking means an FCV, a < means a FMV. These rules are stored in an OWL-DL ontology illustrated in Figure 1(a). Based on this knowledge of rules, the sensors can supply the list of FCVs, sensor_fcv, and FMV, sensor_fmv in the area. Any other vehicle is an unknown vehicle (UV), sensor_uv. The views in Table 2 illustrate the data the sensors provide to the system, and it is the case that we should assume that the classification is unreliable.

The second source of data is provided by a satellite tracking system. The system has live data about all the vehicles it can detect based on their on board tracking devices. The system also has a list of all the FMVs. This knowledge is encapsulated as an OWL-DL ontology illustrated in Figure 2. Any vehicle in the satellite live data which is not on the known lists of FMVs is classified as a UV, satellite_uv, using a dynamic rule. Any vehicle in the satellite live data which is a FMV, is recorded in satellite_fmv. The satellite data does not contain any false positive identifications of FMVs. Table 3 illustrates the live data held by the satellite system and Table 4 illustrates the views over that data feed that classify vehicles.

Based on the sensor and satellite data, a definite list of unknown vehicles can be computed. These vehicles are considered suspicious so a team could be sent to investigate them. A static rule identifies these suspicious rules based on the information from the sensors and the satellite.

sensor_fcv				
X	Y	time	heading	speed
10	15	10:23	sw	40
10	15	11:00	n	90
10	15	14:02	s	35

sensor_fmv				
X	Y	time	heading	speed
10	18	10:25	w	50

sensor_uv				
X	Y	time	heading	speed
10	20	12:14	s	20
10	15	15:00	nw	100

Table 2: Views over the sensor data, identifying friendly civilian (FCV), friendly military (FMV) and unknown (UV) vehicles.

live_data					
TrackingId	X	Y	time	heading	speed
v45	10	15	10:23	sw	40
v46	10	18	10:25	w	50
v52	10	20	10:26	e	30

Table 3: Satellite data feed

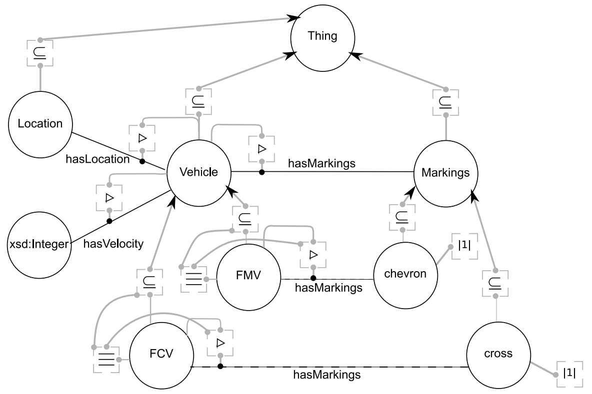
satellite_fmv				
X	Y	time	heading	speed
10	15	10:23	sw	40
10	18	10:25	w	50

satellite_un				
X	Y	time	heading	speed
10	20	10:26	e	30

Table 4: Views over the satellite data

```
class (Location partial owl:Thing)

class (Markings partial owl:Thing)
individual (cross type Markings)
individual (chevron type Markings)

class (Vehicle partial owl:Thing
restriction (hasMarkings someValuesFrom Markings)
restriction (hasLocation someValuesFrom Location)
restriction (hasSpeed someValuesFrom xsd:Integer))

class (FMV complete Vehicle
restriction (hasMarkings hasValue cross))

class (FCV complete Vehicle
restriction (hasMarkings hasValue chevron))
```

(a) OWL-DL

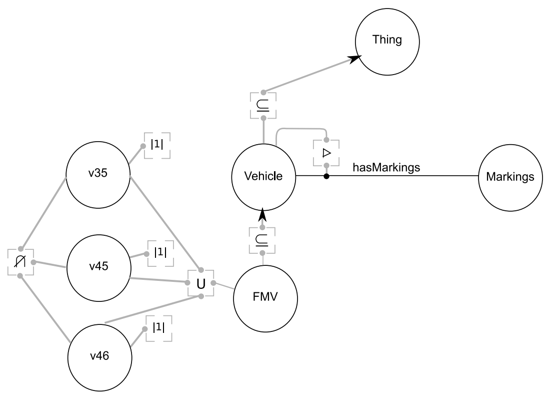


(b) HDM

Figure 1: Knowledge in the sensors

```
class (Vehicle partial owl:Thing
restriction (hasTrackingId someValuesFrom xsd:Integer))
class (FMV complete Vehicle
restriction (oneOf (v35 v45 v46)))

allDifferent (v35 v45 v46)
```

(a) OWL-DL



(b) HDM

Figure 2: Knowledge in the satellite

3 Preliminaries

In common with the previous RoDEx project (CC003) [**?**], we use and extend the **Hypergraph Data Model** (**HDM**) [**?**] as a common data model in which to represent schemas from different knowledge representation languages. Figure 1 shows the OWL knowledge base used for the sensors translated into an HDM graph. Full details of how this is done can be found in [**?**]. We will briefly describe those constructs in the figure and how the specific objects in the figure are derived.

The circles depict the nodes in the graph. They represent some concept in our specific domain of discourse. Nodes can have an extent which is a set of data values from this domain. For example, the circle labelled Vehicle in Figure 1 is a node that represents all the vehicles in our ontology.

The extent of the node is a set of vehicle ids. When discussing specific nodes in the text we write them as node:$\langle\langle nodeName \rangle\rangle$, where $nodeName$ is the label on the node, so the vehicle node will be written as $\langle\langle$Vehicle$\rangle\rangle$. The thick black lines represent the edges. We write edges as follows: edge:$\langle\langle edgeName, objectName_1, \ldots, objectName_n \rangle\rangle$ where $edgeName$ is the label on the edge and $objectName_1, \ldots, objectName_n$ are the nodes or edges that this edge links. The extent of an edge is a tuple whose elements come from the objects it links. In the figure the line linking $\langle\langle$Vehicle$\rangle\rangle$ and $\langle\langle$Markings$\rangle\rangle$ is an edge. We write this as edge:$\langle\langle$hasMarkings, Vehicle, Markings$\rangle\rangle$. Its extent is a tuple, the first element of which comes from the extent of $\langle\langle$Vehicle$\rangle\rangle$ and the second from the extent of $\langle\langle$Markings$\rangle\rangle$.

The grey boxes contain constraint operators. It is sufficient for this paper to give simplified definitions of constraints that assume only atomic values appear in the extent of a node and there are no hyperedges (full definitions that do not make this assumption can be found in [**?**]):

- **mandatory** is represented in diagrams as $s_x \rhd s$ where s_x is a node and s is an edge $\langle\langle \ldots, \mathsf{s_x}, \ldots \rangle\rangle$ which connects it to other nodes, and $Ext_{S,I}(s_x) - \{x \mid \langle \ldots, x, \ldots \rangle \in Ext_{S,I}(s)\} = \emptyset$. In other words, every value of s_x must appear at least once in the edge s.
- **inclusion** is represented in diagrams as $s_1 \subseteq s_2$, and states that $\forall I, Ext_{S,I}(s_1) - Ext_{S,I}(s_2) = \emptyset$, *i.e.* the extent of s_1 is a subset of s_2.
- **cardinality** is represented in diagrams as $|n|$ attached to a node, and implies that $\forall I.|Ext_{S,I}(s)| = n$, *i.e.* the number of instances in s will always be some fixed value n
- **equivalent** is represented in diagrams as $\equiv$ connecting a subset $s_1 \subseteq s_2$ constraint with some other constraints, and has the semantics that if s_2 replaces s_1 in those other constraints, then any data value in $Ext_{S,I}(s_2)$ that obeys the constraints is found in $Ext_{S,I}(S_1)$.

The last two are new constraints we have introduced into the HDM as part of this work. It was necessary to extend our existing constraint set to allow us to represent open world and non-unique name assumption concepts in the HDM.

The grey lines linking the boxes to the nodes and edges show which schema objects the constraints apply to. Constraints do not have an extent, rather they place restrictions on the extents of the objects they are applied to. For example $\langle\langle$Vehicle$\rangle\rangle$ $\rhd$ edge:$\langle\langle$hasMarkings, Vehicle, Markings$\rangle\rangle$ means that for every vehicle in $\langle\langle$Vehicle$\rangle\rangle$ there must be a corresponding tuple in edge:$\langle\langle$hasMarkings, Vehicle, Markings$\rangle\rangle$, *i.e.* every vehicle must have a marking. $\langle\langle$FCV$\rangle\rangle \subseteq \langle\langle$Vehicle$\rangle\rangle$ means every value in $\langle\langle$FCV$\rangle\rangle$ must also appear in $\langle\langle$Vehicle$\rangle\rangle$, *i.e.* every FCV is a vehicle.

4 Representation Issues

4.1 Partial and complete definitions

There are two main approaches to constraints in knowledge representation languages. In the first approach, constraints are used to specify statements that must always hold, and thus they can only partially define the extent of an object. An example of this type is the constraint that specifies that a vehicle must have a location. This constraint is not sufficient to classify an instance as a vehicle. In the second approach, constraints are used to derive the instances of an object. An example of this type of constraints is the constraint that specifies that in order to classify a vehicle as an FMV, then this individual *must* be an vehicle and it must a $<$ marking on it.

All constraints that we have used in the previous RoDEx and AutoMed projects were constraints of the first approach. This type of constraints is available in the relational model, the ER model, RDF/S, OWL-DL, *etc.* In the example that is mentioned above, in the sensor knowledge source there are several constraints of this type. For example, in Figure 1, we can see that there is a partial definition of the Vehicle class, where the constraints specify that a vehicle has an associated location, represented by the mandatory constraint with a mandatory constraint $\mathsf{mandatory}(\langle\langle\mathsf{Vehicle}\rangle\rangle, \langle\langle\mathsf{hasLocation}, \mathsf{Vehicle}, \mathsf{Location}\rangle\rangle)$, a vehicle has an associated velocity, $\mathsf{mandatory}(\langle\langle\mathsf{Vehicle}\rangle\rangle, \langle\langle\mathsf{hasSpeed}, \mathsf{Vehicle}, \mathsf{xsd} : \mathsf{Integer}\rangle\rangle)$, and that a vehicle has an associate marking, $\mathsf{mandatory}(\langle\langle\mathsf{Vehicle}\rangle\rangle, \langle\langle\mathsf{hasMarkings}, \mathsf{Vehicle}, \mathsf{Markings}\rangle\rangle)$.

In order to be able to support the constraints of the second approach, we have to introduce a new constraint primitive: equivalence ($\equiv$). This constraint is available in OWL-DL. For example, in the sensor knowledge source, there are two such constraints defining completely the FMV and FCV classes, as shown in Figure 1. One constraint is used to identify the FMV instances: an individual instance is an FMV if and only if the instance is a vehicle and has a chevron $<$ marking. This rule specifies that FMV is a subclass of Vehicle, $\mathsf{subset}{:}\langle\langle\mathsf{FMV}, \mathsf{Vehicle}\rangle\rangle$, and that any instance of FMV has marking Chevron, $\mathsf{mandatory}(\langle\langle\mathsf{FMV}\rangle\rangle, \langle\langle\mathsf{hasMarkings}, \mathsf{FMV}, \mathsf{Chevron}\rangle\rangle)$. These two constraints are sufficient to derive FMV instances, *i.e.* the FMV class is equivalent to the class where these two constraints hold for all its instances equivalent(FMV,$\langle\langle\subseteq, \mathsf{FMV}, \mathsf{Vehicle}\rangle\rangle$,$\mathsf{mandatory}(\langle\langle\mathsf{FMV}\rangle\rangle, \langle\langle\mathsf{hasMarkings}, \mathsf{FMV}, \mathsf{Chevron}\rangle\rangle)$). The other example is the identification of the FCV instances: an individual instance is an FCV if and only if the instance is a vehicle and has a cross $+$ marking. Similarly to above, we have that equivalent(FMV,$\langle\langle\subseteq, \mathsf{FCV}, \mathsf{Vehicle}\rangle\rangle$,$\mathsf{mandatory}(\langle\langle\mathsf{FCV}\rangle\rangle, \langle\langle\mathsf{hasMarkings}, \mathsf{FCV}, \mathsf{Cross}\rangle\rangle)$).

4.2 Implied edges

In knowledge representation languages, edges are sometimes implied by inheritance. For example, ER relationships of an entity are inherited on the specialisations of that entity. In the HDM, we can define that:

$$\mathsf{edge}{:}\langle\langle \mathsf{e}, \ldots, \mathsf{s_x}, \ldots\rangle\rangle \wedge \mathsf{subset}{:}\langle\langle \mathsf{s_x}, \mathsf{s'_x}\rangle\rangle \\ \rightarrow \mathsf{edge}{:}\langle\langle \mathsf{e}, \ldots, \mathsf{s'_x}, \ldots\rangle\rangle$$

and

$$\mathsf{Ext}(\mathsf{edge}{:}\langle\langle \mathsf{e}, \ldots, \mathsf{s_x}, \ldots\rangle\rangle) \subseteq \mathsf{Ext}(\mathsf{edge}{:}\langle\langle \mathsf{e}, \ldots, \mathsf{s'_x}, \ldots\rangle\rangle)$$

In order to refer to completely edges, we need to define their extent. For example, the $\mathsf{edge}{:}\langle\langle\mathsf{hasMarkings}, \mathsf{FCV}, \mathsf{Cross}\rangle\rangle$ is implied from $\mathsf{edge}{:}\langle\langle\mathsf{hasMarkings}, \mathsf{Vehicle}, \mathsf{Marking}\rangle\rangle$ since $\mathsf{subset}{:}\langle\langle\mathsf{FCV}, \mathsf{Vehicle}\rangle\rangle$ and $\mathsf{subset}{:}\langle\langle\mathsf{Cross}, \mathsf{Marking}\rangle\rangle$, and its extent is:

$$[\{\mathsf{x}, \mathsf{y}\} \mid \\ \{\mathsf{x}, \mathsf{y}\} \leftarrow \mathsf{edge}{:}\langle\langle\mathsf{hasMarkings}, \mathsf{Vehicle}, \mathsf{Marking}\rangle\rangle; \\ \{\mathsf{x}\} \leftarrow \mathsf{node}{:}\langle\langle\mathsf{FMV}\rangle\rangle; \{\mathsf{y}\} \leftarrow \langle\langle\mathsf{Cross}\rangle\rangle]$$

4.3 Individuals

There are multiple ways of representing individual instances in knowledge representation languages.

In one approach, individual instances can be part of the query mappings between schema objects. For example, the following query $[\{x\} \mid \{x, y\} \leftarrow \langle\langle\mathsf{FCV}, \mathsf{Marking}\rangle\rangle; y = \mathsf{Cross}]$ written in IQL refers to the individual Cross. This approach is used in data mappings between relational schemas.

In another approach, individual instances may be explicitly present in the schema representation. This allows more formal definitions, since the instance is no longer some text value. For example, previously we mentioned the constraint $\mathsf{mandatory}{:}\langle\langle\langle\langle\mathsf{FCV}\rangle\rangle, \langle\langle\mathsf{hasMarkings}, \mathsf{FCV}, \mathsf{Cross}\rangle\rangle\rangle\rangle$ which refers to an edge hasMarkings between $\langle\langle\mathsf{FCV}\rangle\rangle$ and another schema object $\langle\langle\mathsf{Cross}\rangle\rangle$, which represents an individual. This approach is used in OWL-DL.

In the RoDEx project, we have used the first approach to represent individual instances. In order to be able to represent individual instances in the schema level, we need to extend our primitive constraints with a cardinality constraint, cardinality. For individual instances their cardinalities must be explicitly specified. Two such constraints are illustrated in Figure 1 for the individuals Cross and Chevron: $\mathsf{cardinality}(\langle\langle\mathsf{Cross}\rangle\rangle,1)$, $\mathsf{cardinality}(\langle\langle\mathsf{Chevron}\rangle\rangle,1)$. In Figure 2 there are three FMV individuals $\langle\langle\mathsf{v35}\rangle\rangle$, $\langle\langle\mathsf{v45}\rangle\rangle$ and $\langle\langle\mathsf{v46}\rangle\rangle$.

5 Static Rules

The are two main distinctions to make in knowledge representation languages. The first is between the open world assumption and the closed world assumption. The second is between the unique name assumption and the non-unique name assumption. After reviewing these distinctions, we discuss how to annotate a rule to illustrate which of these assumptions are being used.

5.1 Open and Closed Worlds

The **open world assumption** (**OWA**) is the assumption that absence of a value from a current data set does not mean that the value does not belong to the data set. It is useful to model systems where there is uncertainty in the data collected, and absense of information does not imply it is false. For example, making markings in the sensor data open world would mean that there being no marking recorded for vehicle id 4 cannot be used to conclude that vehicle id 4 is not an FMV, despite the absence of the required marking for FMVs.

The OWA limits the kinds of inference and deductions an agent can make to those that follow from statements that are known to the agent to be true. The OWA is closely related to the monotonic nature of FOL, *i.e.* adding new information never falsifies a previous conclusion.

The **closed world assumption** (**CWA**) is the assumption that any statement that is not known to be true is false. This is approach is used in relational databases (excluding certain uses of null values), and is appropriate where we are certain of the completeness of our knowledge. For example, it all our FMVs carry a satellite tracking device, we can assume that absence from satellite_fmv implies a vehicle is not an FMV.

Our previous work in RoDEx supported only the CWA. Thus, we need to extend it and provide the ability to annotate schema objects to distinguish between OW and CW objects. In particular, we extend the HDM with a annotating function $R : Nodes \cup Edges \rightarrow \{cw, ow\}$. Function R annotates each node and edge with keywords cw or ow to point whether the default reasoning for the object is closed-world or open-world reasoning, respectively.

The default reasoning in knowledge representation languages differs. For example, OWL supports both OW and CW reasoning, while the relational model supports by default CW reasoning. In our example, the sensor array data do not have complete information about FMV vehicles nor HV vehicles. Sensor data only specify the vehicles that are in a specific area and they are not hiding from the sensors, *i.e.* $R(\langle\langle \mathsf{sensor\_fmv} \rangle\rangle) = ow$. In the satellite knowledge base (Figure 2), the FMV class is a closed world class. It is defined as an enumeration of three FMV individuals, $\langle\langle \mathsf{v35} \rangle\rangle$, $\langle\langle \mathsf{v45} \rangle\rangle$ and $\langle\langle \mathsf{v46} \rangle\rangle$. Any other vehicle is not an FMV if it is not one of the three aforementioned individuals.

In queries, we want to be flexible and be able to change the default reasoning of objects. For example, based on the satellite data we might want to check whether an FMV is in the area using query $member(v35, \langle\langle \mathsf{live\_data} \rangle\rangle^{cw})$, where the $\langle\langle \mathsf{live\_data} \rangle\rangle$ is annotated as CW. Based on the data in Table 3, the answer to this query would be false. If we now take into consideration that some vehicles might not have a very strong signal and therefore they are not detected by the satellite, we might want to use the OWA on the live_data table, *i.e.* $member(v35, \langle\langle \mathsf{live\_data} \rangle\rangle^{ow})$. In this case the query would return back unknown, *i.e.* vehicle v35 might be in the area but cannot be detected.

5.2 Unique name assumption

Regarding the naming assumptions, knowledge representation languages can either support the **unique name assumption** (UNA), *i.e.* that different identifiers always refer to different entities in the world, or the **non-unique name assumption** (nUNA), *i.e.* that different entities in the world might have different identifiers. For this purpose, the HDM needs to be extended and objects need to be annotated to show whether UNA holds for the object or not.

In particular, we extend the HDM with a annotating function $N : Nodes \cup Edges \cup Instances \rightarrow \{UNA, nUNA\}$. Note that instances also need to be annotated, but not *null* values. We remind the reader that the instances of the HDM are $Instances \cup \{null_{unk}\}$, where $null_{unk}$ represents the case when a value should appear but it is currently unknown.

For example, the knowledge we have about vehicles from the sensor array data identify a set of vehicles at different locations at different times. Essentially, vehicles can be identified by their X, Y coordinates, *time*, *heading* and *speed*. However, since the same vehicle can be present at different locations in different times, *e.g.* the vehicle that was at 10:23 and position (10,15) could be the same vehicle that was at position (10,18) 2 minutes later, the knowledge make the nUNA. For example, for the sensor data we have that $N(\langle\langle \mathsf{sensor\_dmv} \rangle\rangle) = nUNA$ and similarly for all its tuples, *e.g.* $N(\langle 10, 15, 10:23, sw, 40\rangle) = nUNA, N(\langle 10, 15, 11:00, n, 90\rangle) = nUNA$, *etc.* In the example, with the satellite data, the knowledge we have is about three FMVs uniquely identified by their tracking id. In OWL-DL, we have explicitly specified that these three vehicles are distinct from one another, using the owl:allDifferent predicate, which translates in the HDM disjoint constraint. Thus, according to the knowledge base in the satellite, the UNA holds for this set of vehicles, *i.e.* $N(\langle\langle \mathsf{FMV} \rangle\rangle) = UNA$.

The UNA and nUNA assumption have implications on queries. Thus, the query language must allow changing the naming assumption made on schema objects, *e.g.* by annotating them. For example, an IQL query asking whether a specific tracking id, *i.e.* a specific vehicle, is present $member(v35, live\_data)$, is translated as $member(v35, live\_data^{una})$ because of the default UNA made in the satellite data. This query for the data illustrated in Figure 3 will return false. However, if we learn that tracking devices have been tampered with, or that our satellite transmission is no longer reliable we need to change the default naming assumption for $\langle\langle \mathsf{live\_data} \rangle\rangle$ into nUNA. The query would then become $member(v35, \langle\langle \mathsf{live\_data} \rangle\rangle^{nuna})$, which returns unknown as a result, since the vehicle with tracking id v35 might be present in the satellite data recorded though with a false tracking id.

5.3 An example of a static rule

There are several examples of static rules in our running example, *e.g.* the definitions of FMV and FCV in the sensor knowledge source (Figure 1) are static rules. Another example of a static rule is the definition of the $\langle\langle \mathsf{sensor\_un} \rangle\rangle$ table:

```
CREATE VIEW sensor_un AS
SELECT x,y,time,heading,speed
FROM sensor_data
WHERE classification IS NULL
```

which as a BAV [?] rule becomes

$$\mathsf{add}(\mathsf{node}{:}\langle\langle \mathsf{sensor\_un} \rangle\rangle, [\{\mathsf{x},\mathsf{y},\mathsf{t},\mathsf{d},\mathsf{v}\} \mid \{\mathsf{x},\mathsf{y},\mathsf{t},\mathsf{c},\mathsf{d},\mathsf{v}\} \leftarrow \mathsf{node}{:}\langle\langle \mathsf{sensor\_data} \rangle\rangle^{\mathsf{cw}}; \mathsf{isnull}(\{\mathsf{c}\})])$$

The sensor_data table is annotated as CW rather than OW because a vehicle with id=4 would not be in the sensor_un table if the table is annotated as OW, because even though there is a tuple in the sensor_data table that shows that this vehicle does not have a classification, the OWA allows the existence of another tuple which is not yet recorded and where the specific vehicle *has* a classification.

6 Dynamic Rules

Dynamic rules (or active rules) are rules that allow changes in the data to cause other changes in the data to occur. To illustrate the nature if this type rule, consider the following two SQL triggers used in the satellite system to update its satellite_fmv and satellite_un tables based on its live data.

The update_satellite_fmv trigger identifies all vehicles in the live data which are FMVs, according to its internal list of FMV vehicles, stored in the fmv table. The update_satellite_un trigger identifies all vehicles in the live data which are not FMVs, according to the fmv table, and stores in the the satellite_un table as unknown vehicles.

```
CREATE TRIGGER update_satellite_fmv AFTER INSERT ON live_data

  SELECT INTO TrackingId,X,Y,time,heading,speed
  FROM live_data,fmv
  WHERE live_data.TrackingId=fmv.TrackingId

  INSERT INTO satellite_fmv
  VALUES (TrackingId,X,Y,time,heading,speed)

CREATE TRIGGER update_satellite_un AFTER INSERT ON live_data

  SELECT INTO TrackingId,X,Y,time,heading,speed
  FROM live_data
  WHERE TrackingId NOT IN (SELECT TrackingId FROM fmv)

  INSERT INTO satellite_un
  VALUES (TrackingId,X,Y,time,heading,speed)
```

Such dynamic behaviour can be represented using executable temporal logic rules [?]. Rules in executable temporal logic take the form $\mathsf{declarative\ past} \rightarrow \mathsf{executable\ future}$, and have the semantics that if at any time the query declarative past holds, then executable future is made to hold. For example, to represent the first of the triggers listed above we would have rules of the form:

$\{\mathsf{I},\mathsf{t}\} <\!- \mathsf{edge}{:}\langle\!\langle \_, \mathsf{live\_data}, \mathsf{TrackingId}\rangle\!\rangle;$
$\mathsf{not}\ \bullet\{\mathsf{I}\} <\!- \mathsf{node}{:}\langle\!\langle \mathsf{live\_data}\rangle\!\rangle; \{\mathsf{I}\} <\!- \mathsf{node}{:}\langle\!\langle \mathsf{fmv}\rangle\!\rangle \rightarrow$
$\quad \bigcirc\{\mathsf{I},\mathsf{t}\} <\!- \mathsf{edge}{:}\langle\!\langle \_, \mathsf{satellite\_fmv}, \mathsf{TrackingId}\rangle\!\rangle$
$\{\mathsf{I},\mathsf{x}\} <\!- \mathsf{edge}{:}\langle\!\langle \_, \mathsf{live\_data}, \mathsf{X}\rangle\!\rangle;$
$\mathsf{not}\ \bullet\{\mathsf{I}\} <\!- \mathsf{node}{:}\langle\!\langle \mathsf{live\_data}\rangle\!\rangle; \{\mathsf{I}\} <\!- \mathsf{node}{:}\langle\!\langle \mathsf{fmv}\rangle\!\rangle \rightarrow$
$\quad \bigcirc\{\mathsf{I},\mathsf{t}\} <\!- \mathsf{edge}{:}\langle\!\langle \_, \mathsf{satellite\_fmv}, \mathsf{X}\rangle\!\rangle$
$\vdots$

where $\bullet$ means 'in the previous moment', and hence $\{\mathsf{I},\mathsf{t}\} <\!- \mathsf{edge}{:}\langle\!\langle \_, \mathsf{live\_data}, \mathsf{TrackingId}\rangle\!\rangle;$ $\mathsf{not}\ \bullet\{\mathsf{I}\} <\!- \mathsf{node}{:}\langle\!\langle \mathsf{live\_data}\rangle\!\rangle$ identifies new instances of the live_data table, and the TrackingId associated with it.

7 Conclusion

In this paper we have outlined our work in building a common representation of knowledge bases. The three main areas we have dealt with are the differences between the open and closed world assumptions, the differences between using unique name and non-unique name assumptions, and the modelling of the temporal aspect of the knowledge base. This common representation will serve as a platform for our later work on knowledge integration.

Acknowledgements

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.

References

[1] H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future.* Research Studies Press, Somerset, 1996.

[2] M. Boyd and P. McBrien. Comparing and transforming between data models via an intermediate hypergraph data model. *Journal on Data Semantics*, IV:69–109, 2005.

[3] D. Le, P. McBrien, and A.C.Smith. RoDEx: Robust data exchange for unreliable networks. In *Proc. 3rd SEAS DTC Technical Conference*, 2008.

[4] P. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE'03*, pages 227–238. IEEE, 2003.

[5] A. Poulovassilis and P. McBrien. A general formal framework for schema transformation. *Data and Knowledge Engineering*, 28(1):47–71, 1998.

[6] A. Sheth and J. Larson. Federated database systems. *ACM Computing Survers*, 22(3):183–236, 1990.

[7] A. Smith, N. Rizopoulos, and P. McBrien. Rokex: Report of work package 1. Technical report, Imperial College London, August 2008.