

# CC013 - Robust application-layer Knowledge Exchange protocols for networks of semi-autonomous vehicles

---

Nikos Rizopoulos, Andrew Smith and Peter McBrien  
Dept. of Computing  
Imperial College London



**BAE SYSTEMS**

**Imperial College**  
London

# Overview

---

- ρ Objectives
- ρ Representing Knowledge
- ρ Translating and Integrating Knowledge
- ρ Current and Forthcoming Work

# Objectives

---

- ⌘ **Extend capabilities of RoDEx (CC003) to incorporate knowledge components**
  - ⌘ **Different assumptions about knowledge held in knowledge bases (KBs)**
  - ⌘ **Rules**
  - ⌘ **Provenance of decisions and data and knowledge quality**
- ⌘ **Develop tools that allow us to apply these techniques to mission planning and other tasks, with UXVs**

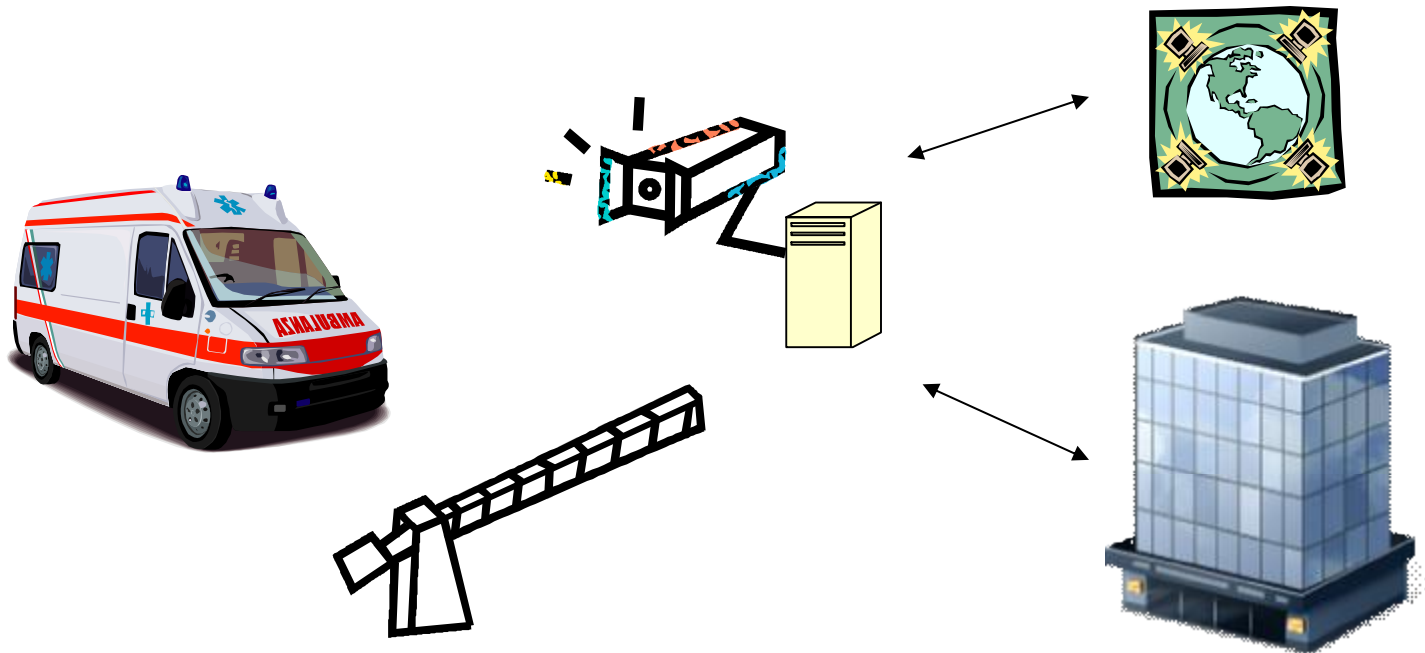
# Collaborations

---

- ρ Cooperate with group from Roke on IF058 and PPEM016
- ρ SER011
- ρ SIG Autonomous Information Management

# A Knowledge Integration Scenario

---



**enemyMarking(m) <- Marking(m) & m = 'Enemy1'**  
**allowedToEnterVehicle(x) <- hasMarking(x,m) & not enemyMarking(m)**

# Assumptions

---

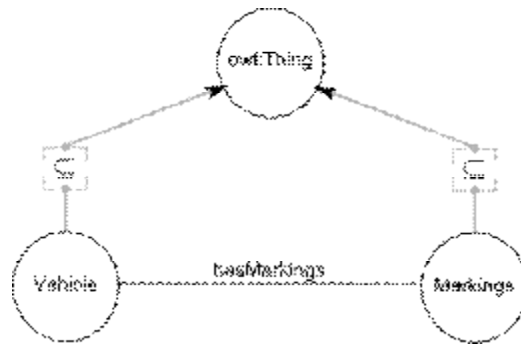
- ⌘ Closed World Assumption (CWA) vs Open World Assumption (OWA)
  - ⌘ CWA - what is not currently known to be true, is false
  - ⌘ OWA – the truth-value of a statement is independent of whether or not it is *known* by any single observer or agent to be true
- ⌘ The Unique Name Assumption (UNA)
  - ⌘ all objects with different names always refer to different entities in the world

# RoKEx Methodology I

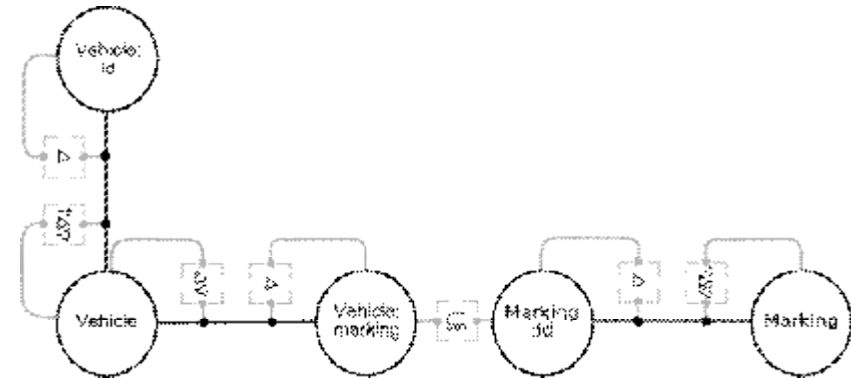
---

- p** Create a Common Knowledge Model (CKM) capable of representing schemas from a wide range of knowledge modelling languages
  - n** able to represent different assumptions
  - n** measures for knowledge quality

# RoDEx Representation of OWL and SQL schemas



OWL-DL



SQL

# Representing the Different Assumptions in HDM

---

**p**  $R$ : Node U Edges  $\rightarrow$  {cw,ow}

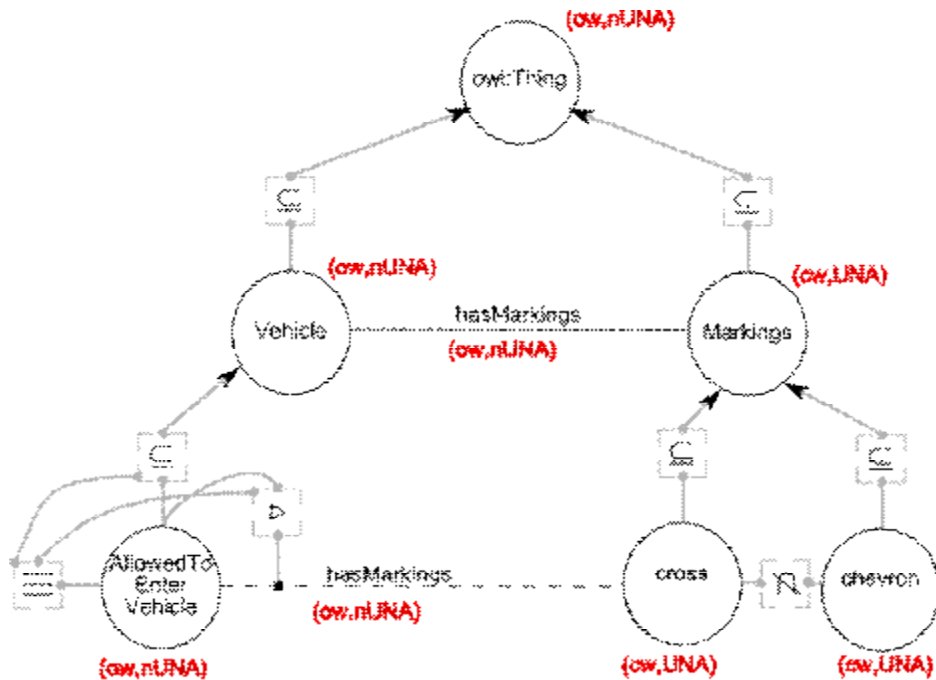
**n**  $R(\text{node: } \langle \langle \text{Marking} \rangle \rangle) = \text{cw}$

**p**  $N$ : Nodes U Edges U Instances  $\rightarrow$   
{UNA,nUNA}

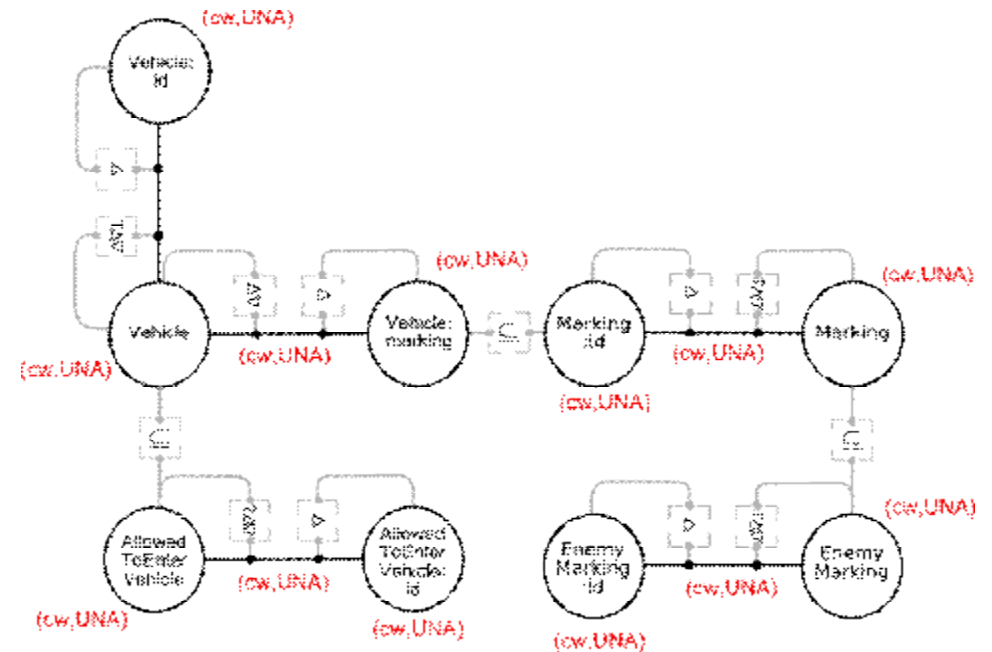
**n**  $N(\text{Inst: } \langle \text{Vehicle} \rangle) = \text{nUNA}$

**n**  $N(\text{Inst: } \langle \text{'cross'} \rangle) = \text{UNA}$

# RoKEx representation of OWL and SQL schemas



OWL  
(Derived/Named  
Classes)



SQL  
(Views)

# RoKEx Methodology II

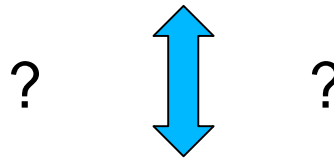
---

- ⌘ Define a mapping language that allows us to translate and combine information from both types of system
  - ⌘ mechanisms for tracking provenance

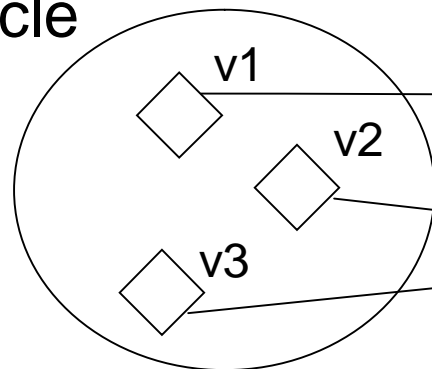
# Translation between SQL and OWL

vehicle	
<u>vehicle_id</u>	marking
v1	UNMarking
v2	NatoMarking
v3	USMarking

SQL

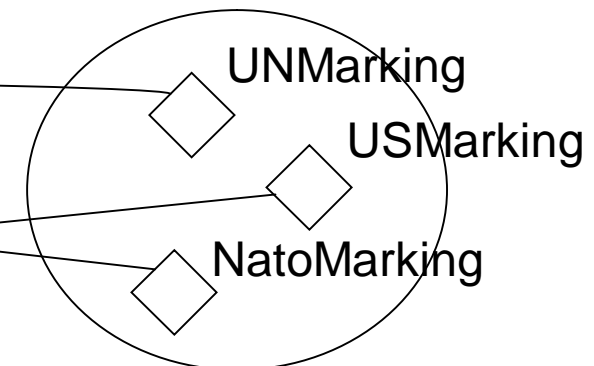


Vehicle



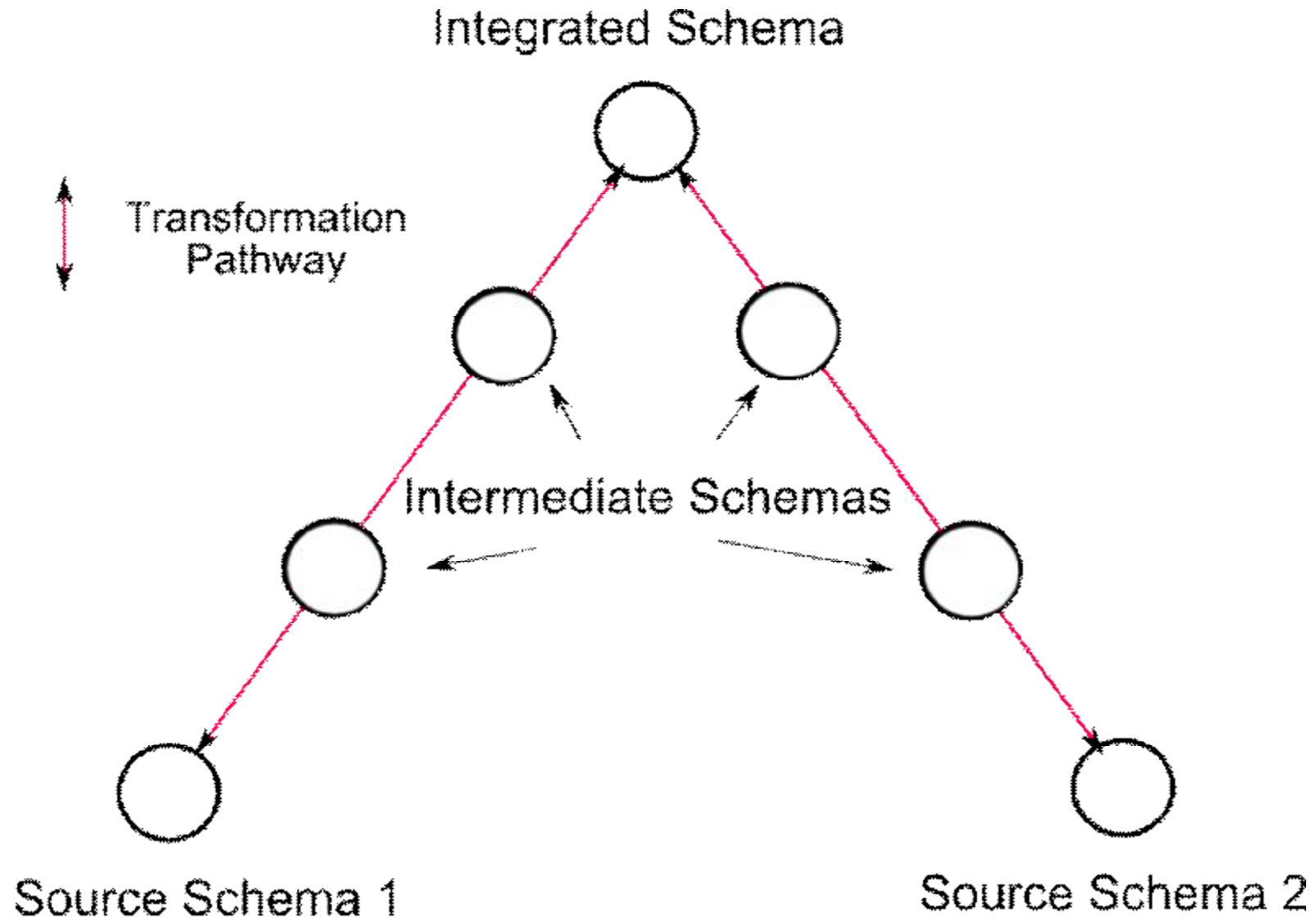
hasMarking

Marking



OWL-DL

# AutoMed Transformation Pathways (BAV)

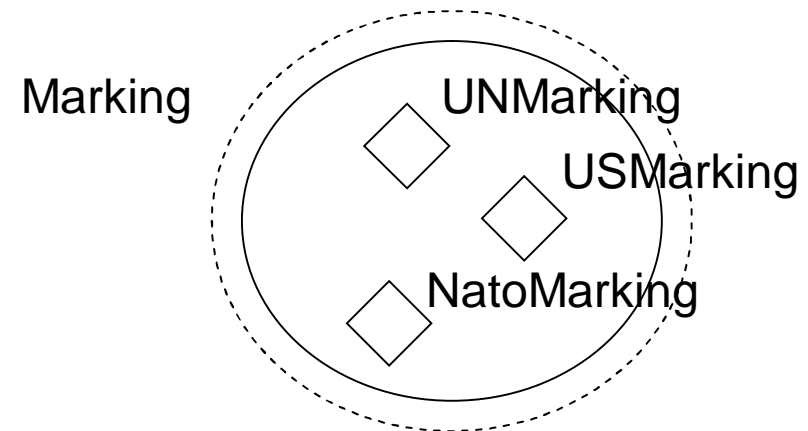
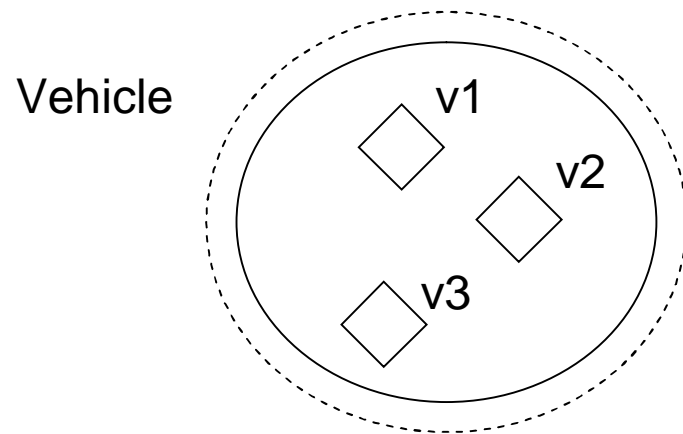


# CWA in OWL

## p Enumerated classes

n owl:class:Vehicle {v1, v2, v3}

n owl:class:Marking {UNMarking, NatoMarking, USMarking}



# OWA in SQL

---

- ⌘ Using NULL values and 3-valued logic
- ⌘ Is v4 a member of Vehicle?
  - ⌘ CWA: *false*
  - ⌘ OWA: *unknown*

Vehicle	
<u>vehicle_id</u>	marking
v1	UNMarking
v2	NatoMarking
v3	USMarking
NULL	NULL

# UNA: SQL to OWL

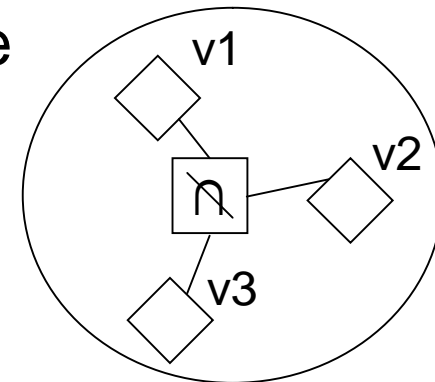
## p Translation from SQL to OWL

n owl:AllDifferent, owl:differentFrom

Vehicle
<u>vehicle_id</u>
v1
v2
v3

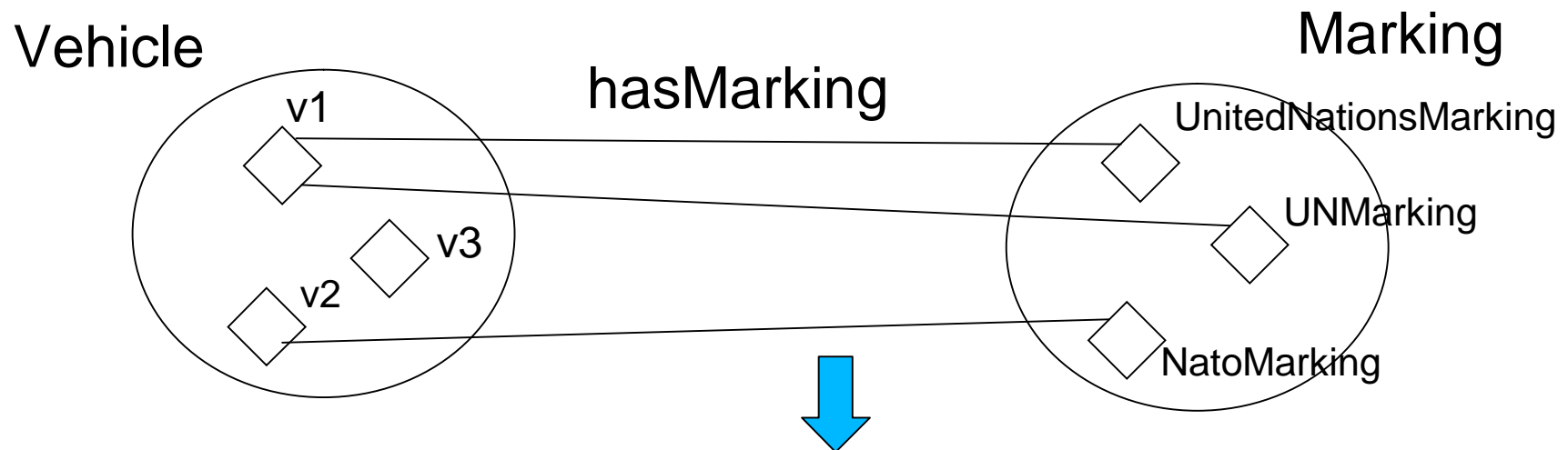


Vehicle



# UNA: OWL to SQL

- Identify the subset of the ontology where UNA holds
- Use aliases where the UNA does not hold



Vehicle	
<u>vehicle_id</u>	marking
v1	UnitedNationsMarking
v2	NatoMarking

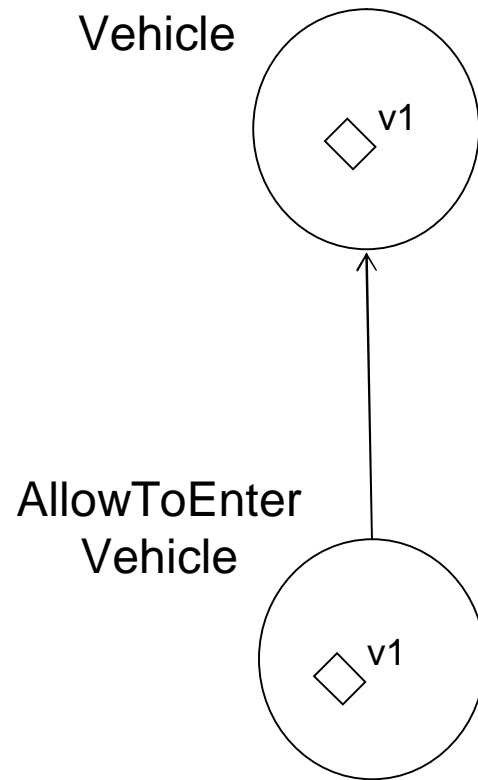
Marking_alias	
<u>marking</u>	alias
UnitedNationsMarking	UNMarking

# Active Rules

---

- ⌘ SQL has constraints to prevent insertions/updates
- ⌘ OWL-DL uses reasoning to infer new instances

# Simulating OW reasoning in SQL



OWL-DL Reasoning



```
CREATE FUNCTION atevehicle_implications() RETURNS  
TRIGGER AS 'BEGIN  
  INSERT INTO Vehicle VALUES(NEW.vehicle_id);  
  
  IF (NEW.vehicle_id) IN  
    (SELECT vehicle_id FROM AllowToEnterVehicle) THEN  
    RETURN NULL;  
  ELSE  
    RETURN NEW;  
  END IF;  
END'  
LANGUAGE plpgsql;  
  
CREATE TRIGGER atevehicle_implications  
BEFORE INSERT ON AllowToEnterVehicle FOR EACH ROW  
EXECUTE PROCEDURE atevehicle_implications();
```

SQL Trigger

# Simulating Constraints in OWL-DL

---

- ⌘ Use enumerated types and UNA techniques
- ⌘ This will lead to inconsistent OWL-DL KBs when a reasoner is run

# Current Work

---

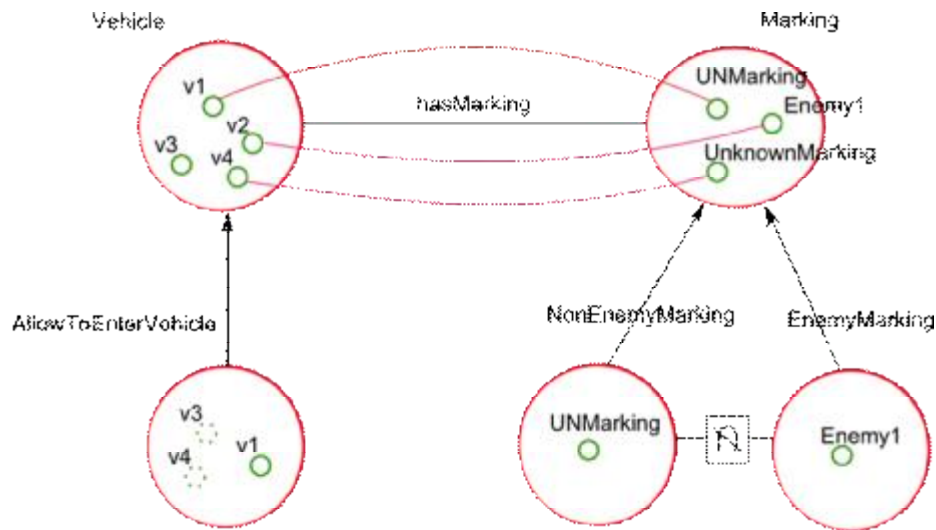
- ρ Integrating knowledge base objects that contain the same knowledge but make different assumptions about it
- ρ Integrating Rules – AND and OR
- ρ Quality and Provenance
- ρ Dynamic Rules - ECA
- ρ Export of Integrated Schema

# Thank You!

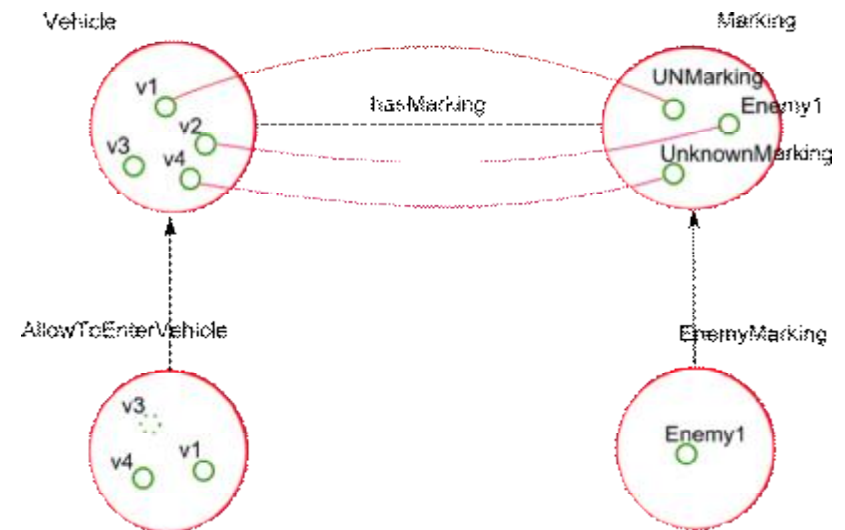
---

**p Q & A**

# OWA vs CWA



○ Might be in the set  
● is in the set  
↑ Subset



○ Might be in the set  
● is in the set  
↑ Subset

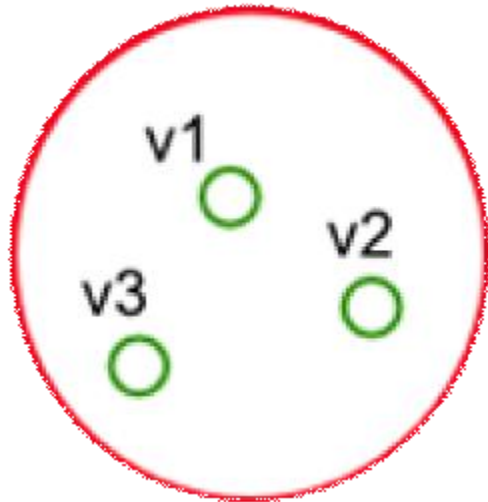
Open World  
(OWL-DL)

Closed World  
(SQL)

# UNA vs No UNA

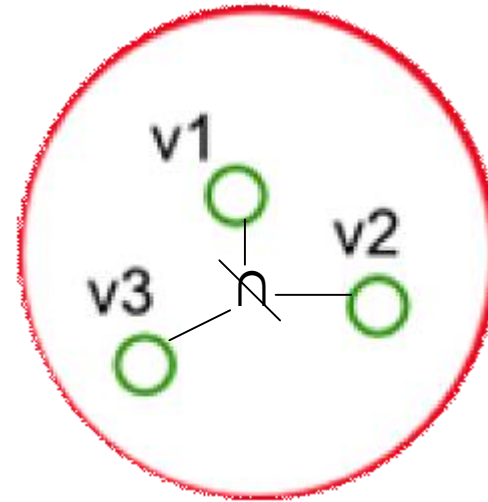
---

Vehicle



No UNA  
(OWL-DL)

Vehicle



UNA  
(SQL)

# Dynamic Rules

---

- ⌞ Based on executable temporal logic
- ⌞ Rules allow us to specify what happens at different points in time
  - {v,m} ← edge: <<\_, Vehicle, cross>> →
    - {v} ← <<AllowToEnterVehicle>>
- ⌞ Allow us to model ECA type rules

# Exporting the Integrated Schema

---

- ⌘ **Modified RoDEx wrappers allow materialisation of integrated schema in OWL-DL or SQL**