A multi-agent approach towards cooperative overtaking in vehicular networks

Adrian Groza
Intelligent Systems Group
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Adrian.Groza@cs.utcluj.ro

Bogdan Iancu
Networks Laboratory
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Bogdan.Iancu@cs.utcluj.ro

Anca Marginean
Intelligent Systems Group
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Anca.Marginean@cs.utcluj.ro

ABSTRACT

This paper deals with the integration of agent technology in the emerging field of vehicular networks. The agents are empowered with domain knowledge and they can perform geospatial and temporal reasoning. For the domain knowledge we developed a vehicular network ontology. The geospatial reasoning is performed with AllegroGraph, while event reasoning in RacerPro. The vehicle overtaking scenario is used to exemplify our solution.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Agents; C.2.1 [**Network Architecture and Design**]: Network communications

Keywords

agents, vehicular networks, ontologies, cooperation, temporal reasoning, spatial reasoning.

1. INTRODUCTION

Vehicular Ad-hoc Networks (VANETs) are one of the most promising applications of mobile ad-hoc networks. VANETs use vehicles as mobile nodes able to self-configure in order to create a communication network via wireless links. The communication takes place both between vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I), aiming to enable *automated* cooperation among different vehicles on the road.

Vehicle-2-X technology has just solved low level aspects with respect to ad-hoc networks or regulatory norms and standards. Thus, much work remains at the application layer, to seamlessly integrate the newly developed services. In this line, innovative communication and cooperative techniques are needed to model different interaction patterns. Our hypothesis is that, VANET-related applications could benefit from the current advances in multi-agent technology.

WIMS '14, June 02-04 2014, Thessaloniki, Greece
Copyright 2014 ACM 978-1-4503-2538-7/14/06...$15.00.
http://dx.doi.org/10.1145/2611040.2611096.

Table 1: KRSS syntax and semantics of $\mathcal{ALC}$.

Constructor	*Syntax*	*Semantics*
negation	`(not C)`	$\Delta^I \setminus C^I$
conjunction	`(and C D)`	$C^I \cap D^I$
disjunction	`(or C D)`	$C^I \cup D^I$
existential restriction	`(some r C)`	$\{x \in \Delta^I \mid \exists y : (x, y) \in r^I \wedge y \in C^I\}$
value restriction	`(all r C)`	$\{x \in \Delta^I \mid \forall y : (x, y) \in r^I \rightarrow y \in C^I\}$
individual assertion	`(instance a C)`	$\{a\} \in C^I$
role assertion	`(related a b r)`	$(a^I, b^I) \in r^I$

In this paper we exploit the *communication and reasoning capabilities of multi-agent systems* to deploy cooperative communication in vehicular networks. Our solution aims to increase situation awareness of vehicles by performing continuous reasoning and geospatial reasoning on topological maps and exchanging structured knowledge among agents about various traffic events.

Our contributions are: 1) formalizing vehicular messages and the corresponding vehicular data in description logic; 2) defining agent types for the overtaking vehicular scenario; 3) use of agent technology to continuously collect context information (vehicle direction, available communication channels, etc.); 4) employ reasoning on vehicular ontologies to increase situation awareness and event recognition.

The remaining of the paper is structured as follows: Section 2 presents the technologies that we employ: description logic and vehicular networks. In section 3 we model VANETs-related knowledge and agents in description logic. Section 4 shows how geospatial reasoning is performed on the above knowledge. Section 5 introduces the running scenario. Section 6 discusses the related work, while section 7 concludes the paper.

2. ENACTED TECHNOLOGIES

The agents reason on knowledge represented in *description logic* (DL) and they use *vehicular communication* to increase situation awareness.

2.1 Description logic

In the description logic $\mathcal{ALC}$, concepts are built using the set of constructors formed by negation, conjunction, disjunction, value restriction, and existential restriction [2], as shown in table 1 in the Knowledge Representation System Specification (KRSS) syntax. Here, C and D represent concept descriptions, while r is a role name. The semantics are

```
1. (in-tbox Vanet)
2. (define-primitive-role belongsTo :domain Vehicle
       :range (or Individual Company PublicAgency))
3. (implies PrivateVehicle Vehicle)
4. (implies PublicVehicle Vehicle)
5. (implies Bus PublicVehicle)
6. (implies Police PublicVehicle)
7. (implies PublicVehicle (all belongsTo PublicAgency))
8. (implies LocalTransportAgengy PublicAgency)
9. (implies RoadSideUnit (some belongsTo (or PublicAgency
                                             PrivateServiceOp)))
```

Figure 1: Modeling VANET terminology in DL.

```
10. (in-abox vanet-salonic Vanet)
11. (instance b1 Bus)
12. (instance lta-salonic LocalTransportAgengy)
13  (instance rsu1 RoadSideUnit)
14. (related b1 lta-salonic belongsTo)
15. (related rsu1 lta-salonic belongsTo)
```

Figure 2: Modeling assertions in VANETs.

defined based on an interpretation $I = (\Delta^I, \cdot^I)$, where the domain Δ^I of I contains a non-empty set of individuals, and the interpretation function $\cdot^I$ maps each concept name C to a set of individuals $C^I \in \Delta^I$ and each role r to a binary relation $r^I \in \Delta^I \times \Delta^I$. The last column of table 1 shows the extension of $\cdot^I$ for non-atomic concepts.

An ontology consists of terminologies (or TBoxes) and assertions (or ABoxes).

Definition 1. *A terminology TBox is a set of terminological axioms of the form (equiv C D) or (implies C D), where C and D are concepts.*

Example 1 (Terminological box). *In the tbox `Vanet` in figure 1, the vehicles are partitioned into private (belonging to individuals or private companies in line 2) and public (i.e. buses, police in lines 5-6). The axiom 7 specifies that a `PublicVehicle` should belong only to public agencies. Road side units can belong to the government or private service operators (axiom 9).*

Definition 2. *An assertional box ABox is a finite set of concept assertions (instance a C) or role assertions (related a b r), where C designates a concept, r a role, and a and b are two individuals. Usually, the unique name assumption holds within the same ABox.*

Example 2 (Assertional box). *The assertional box `vanet-salonic` makes use of the terminologies in the `Vanet` tbox (line 10 in figure 2). The bus `b1` (line 11) belongs to the local transportation agency `lta-salonic` (line 14). Similarly, the road side unit `rsu1` (line 13) operates under the same public agency `lta-salonic` (line 15).*

A concept C is satisfied if there exists an interpretation I such that $C^I \neq \emptyset$. The concept D subsumes the concept C, represented by `(implies C D)`, if $C^I \subseteq D^I$ for all interpretations I. Constraints on concepts (i.e. `disjoint`) or on roles (`domain`, `range` of a role, `inverse` roles, `transitive` properties), number contraints (`max`, `min`), role inheritance (`parent` role) can be specified in more expressive description logics [1].

2.2 Vehicle-2-X communication

Communication among vehicles relies on the Wireless Access in Vehicular Environments (WAVE) protocol and the IEEE 802.11p wireless standard. The WAVE architecture provides interoperable wireless V2V and V2Infrastructure services. The IEEE 1609 family of standards (1609 WG - Dedicated Short Range Communication Working Group) defines the WAVE architecture and details different management and security activities required for proper operation of the applications. The standards provide a baseline for developing a wide range of applications for a variety transportation services: safety, traffic, road conditions, infotainment and many others.

Two types of messages enable V2X communication [11]: periodic messages and event-driven messages. A vehicle advertises its current status to other vehicles, for example, vehicle position, speed, direction of travel by sending periodic messages in the network. Event-driven messages are described as warning messages usually sent to other vehicles if hazardous situations are detected.

Geocast ad hoc routing protocol is introduced as the core networking protocol for vehicular communication based on IEEE 802.11 technology promoted by the Car-to-Car Communication Consortium (C2C-CC) in Europe [8]. Geocast protocol presumes that vehicles have information regarding their geographic location through GPS.

The three key components of Geocast protocol are beaconing, location service and forwarding. Beaconing allows nodes to continuously and periodically broadcast information to all neighbors in the reception range, to permit cooperative awareness. The location service can map a node's ID to its geographical position. Forwarding enables relaying data through VANET network to a certain destination, using GeoUnicast - provides data delivery between two nodes via multiple wireless hops, or GeoBroadcast - distributes data packets by flooding, where nodes re-broadcast the packets if they are located in the geographical region determined by the packets.

3. MODELING KNOWLEDGE FOR VEHICULAR AGENTS

In this section we: i) formalise various data elements from vehicular networks in description logic, ii) define specific vehicular messages in description logic, and iii) introduce different types of agents that can be defined in context of vehicular applications.

3.1 Types of data

The data elements communicated in vehicular networks are formalized by the Society of Automotive Engineers (SAE) standard. We start by modeling this vocabulary in our vehicular ontology.

Primitive data elements such as `Latitude, Longitude, Velocity, VehicleSize` (lines 16-19 in figure 3) are usually packed in a `DataFrame` to optimise the communication. In lines 20-22 the `Latitude` is further defined as having a value, a unit of measure, and an accuracy of measurement. The concept `DataFrame` should have a unique frame `ID`, a textual description, and several primitive data elements (lines

[1]We provide only some basic terminologies of description logics in this paper to make it self-contained. For a detailed explanation about families of description logics, the reader is referred to [2].

```
16. (implies Latitude PrimitiveDataElement)
17. (implies Longitude PrimitiveDataElement)
18. (implies Velocity PrimitiveDataElement)
19. (implies VehicleLength PrimitiveDataElement)
20. (implies Latitude (and (some hasValue Real)
21.                         (all measures UnitOfMeasure)
22.                         (some hasAcc Real)))
23. (implies DataFrame (and (some hasID ID)
24.                    (some hasDescription String)
25.                    (some hasContent PrimitiveDataElement)))
26. (implies PositionDataFrame (and DataFrame (equal hasID 21)
27.      (some hasLat Latitude) (some hasLong Longitude)))
28. (implies SenderDataFrame (and DataFrame (equal hasID 15)
29.            (some hasLength Real) (some hasWidth Real)
30.            (some hasModel Vehicle)))
31. (parent-role hasLatitude hasData)
32. (parent-role hasLongitude hasData)
33. (parent-role hasLength hasData)
34. (parent-role hasWidth hasData)
```

Figure 3: The Tbox of data elements.

```
35. (instance lat1 (and Latitude (= hasValue 40.6393)
                 (= hasAcc .2)))
36. (instance long1 (and Longitude (= hasValue 22.9446)
                  (= hasAcc .2)))
37. (instance p1 (and PositionDataFrame
38.              (= hasLatitude lat1) (= hasLongitude long1)))
39. (instance daciaLogan Vehicle)
40. (instance s1 (and SenderDataFrame
41.              (= hasLength 4.288) (= hasWidth 1.989)))
42.              (equals hasModel daciaLogan)))
```

Figure 4: The Abox for data elements.

23-25). To reduce communication overhead, we packed the latitude and longitude into a *data frame* called `PositionDataFrame` (lines 26-27). The `SenderDataFrame` introduced in lines 28-30 is used to include specifications about the vehicular agent that sends the message. The terminologies in figure 3 are instantiated in figure 4. The position `p1` is defined by the coordinates `lat1` and `long1` by the assertions 35 and 36. An instance of the `SenderDataFrame` appears in lines 40-42. The primitive data elements of the frame specify the length and width of the model `daciaLogan`, which is an instance of `Vehicle` (line 39).

For each vehicular application, part of the SAE standard is enacted. Consider the overtaking scenario. The following data are relevant: message ID, heading, speed, acceleration, vehicle length, position, direction. These data elements need to be gathered and compressed in one packet or message to be sent to nearby vehicles.

3.2 Types of messages

We represent low level aspects of VANET communication in order to facilitate agents to reason on the messages at the application level.

DEFINITION 3. *A vehicular message is a tuple* $\langle CT, TM, Data, Range\rangle$, *where CT is a shortcut for the concept CommunicationType, TM transmission mode, Data represents the content of the message, and Range is maximum communication range.*

The corresponding definition of the `Message` concept in DL appears in lines 51-54 of figure 5. The "n-ary relationship" ontology design pattern [13] was used to define the `Message` concept. The `Data` element includes either data frames or primitive data elements (axiom 55-56).

```
51. (implies Message (and (some hasComm CommunicationType)
52.                  (some hasTransmission TransmissionMode)
53.                  (some hasContent Data)
54.                  (some hasRange Integer)))
55. (implies Data (or (some hasContent DataFrame)
56.               (some hasContent PrimitiveDataElement)))
57. (equiv CommunicationType (or V2V V2I))
58. (disjoint V2V V2I)
59. (equiv TransmissionMode (or Periodic EventDriven))
60. (disjoint Periodic EventDriven)
61. (implies PeriodicMessage (and Message
62.                          (some hasTransmission Periodic)
63.                          (some hasfrequency Time)))
64. (implies EventDrivenMessage (and Message
65.                         (some hasTransmission Event-Driven)
66.                         (some isTriggeredby Event)))
67. (implies Accident Event)
68. (implies TrafficJam Event)
69. (implies Overtaking Event)
70. (ShortRangeMessage (and Message (max hasRange 1000)))
```

Figure 5: Vehicular communication in description logic.

The concepts `CommunicationType` (57-58) and `TransmissionMode` (59-60) are modeled with the partition ontology design pattern [13]. Periodic messages (lines 61-63) are generated to inform nearby vehicles about the current status of the vehicle (i.e. speed, position, direction). These messages are broadcasted periodically (i.e. 100 ms). Event-driven messages (lines 64-66) are triggered by various traffic events (i.e. unsafe situation). Such a message contains the location of the vehicle, the time, and the event type (i.e., `Acccident`, `TrafficJam`, or `Overtaking` in axioms 67-69). More specific messages can be defined based on the generic concept `Message` in lines 51-54. For short range messages, the range of communication using dedicated short-range communications (DSRC) is up to 1000 m (definition 70).

EXAMPLE 3. *A lane change warning message takes place between several vehicles and it is triggered by an event. The data element contains the SenderDataFrame, the PositionDataFrame, and also primitive data like velocity, acceleration, turn signal status. The message is designated for vehicles in an area below 400m.*

```
(implies LaneChangeWarningMessage
  (and (some hasComm V2V)
       (some hasTransmission EventDriven)
       (some hasDataFrame SenderDataFrame)
       (some hasDataFrame PositionDataFrame)
       (some hasPrimitiveDataElement Velocity)
       (some hasPrimitiveDataElement Acceleration)
       (some hasPrimitiveDataElement TurnSignalStatus)
       (some hasRange 400)))
```

The instantiation of a specific `LaneChangeWarningMessage` is illustrated in figure 6. In the first part (lines 71-79), the elements of the message are instantiated. In the second part (lines 80-85), these elements are asserted to the individual `lc-message` of type `LaneChangeWarningMessage`. The data frame list `dfs` includes the `SenderDataFrame s1` previosly defined (lines 40-42) and the `PositionDataFrame p1` (lines 37-38 in figure 4). In lines 76-79, the individual `pdes` encapsulates volatile data about primitive data elements velocity, acceleration, and turn signal status. Based on the definition 70, the agents are able to infer that the `lc-message` is also an instance of the class `ShortRangeMessage`.

```
71. (instance c2c V2V)
72. (instance laneChanging Event)
73. (instance dfs DataFrames)
74. (related dfs s1 hasDataFrame)
75. (related dfs p1 hasFataFrame)
76. (instance pdes (and PrimitiveDataElements)
77.                (= velocity 70)
78.                (= acceleration 1.3)
79. (equal turnSignalStauts true)))
80. (instance lc-message (and LaneChangingWarningMessage
81                        (= hasRange 400)))
82. (related lc-message c2c hasComm)
83. (related lc-message laneChanging hasEvent)
84. (related lc-message dfs hasDataFrames)
85. (related lc-message pdes hasPrimitiveDataElements)
```

Figure 6: A specific lane changing message.

```
91. (implies PassiveAg (and Agent (some sendMsg PeriodicMsg)))
92. (implies ActiveAg (and Agent (some sendMsg EventDrivenMsg)))
93. (implies NormalAg (and Agent (some hasEvent NormalOvertaking)))
94. (implies FlyingAg (and Agent (some hasEvent FlyingOvertaking)))
95. (implies PiggyAg (and Agent (some hasEvent PiggyOvertaking)))
96. (implies NormalAg (and Agent (some hasEvent NormalOvertaking)))
97. (implies Two+Ag (and Agent (some hasEvent TwoPlusOvertaking)))
98. (implies PoliteAg (and Agent
99.     (or (some hasEvent DecreasingSpeedDuringOvertaking))
100.        (some hasEvent SignalsRightBeforeOvertaking)
101.        (some hasEvent ThankMsgAfterLaneChanging))))
```

Figure 7: Types of agents.

3.3 Types of agents

In case of the overtaking maneuver, the vehicular agents can be defined based on several vectors: i) transmission mode, ii) type of the overtaking maneuver, and iii) politeness.

Depending on the transmission mode, there are two types of cooperative agents: passive cooperative and active cooperative.

A *passive cooperative* agent broadcasts its position, velocity, direction, etc. with a specific frequency (line 91 in figure 7). The agents in the same VANET continuously reason on the received data to identify various traffic events.

An *active cooperative* agent (line 92) broadcasts a specific message when it recognizes a traffic event, in order to warn vehicles in its neighborhood. The message is sent towards the closest vehicular agent (VA), towards the closest road side unit (RSU), towards the agents within a specific range, or towards the entire VANET in which the agent belongs.

Depending on the overtaking maneuver, we can define four agents: normal, flying, piggy, and 2+ (axioms 93-97). A `NormalAg` follows a vehicle and waits for a safe sufficient gap to perform an overtaking maneuver. A `FlyingAg` does not adjust its speed to the speed of the overtaken vehicle, but continues at its current speed when overtaking. The `PiggyAg` follows another vehicle that overtakes a slower vehicle, while the `Two+Ag` type overtakes two or more vehicles.

From the politeness point of view (lines 98-101), a `PoliteAg` decreases its speed in order to facilitate the overtaker to come back in the initial line. Also, in real situations, polite drivers use to signal right in order to signal to the vehicle behind that he will be facilitating the overtaking. In a vehicular context, this is modeled by an acknowledge message when the vehicle behind conveys its intention to overtake.

```
102. (in-tbox EmergencyVehicles)
103. (define-role onSameStreet :inverse onSameStreet
104.                :domain Vehicle :range Vehicle)
105. (implies Bus Vehicle)
106. (implies EmergencyVehicle Vehicle)
107. (implies Ambulance EmergencyVehicle)
108. (disjoint Bus EmergencyVehicle)
109. (parent-role inFrontOf onSameStreet)
110. (in-abox vanet-salonic)
111. (instance a Ambulance)
112. (instance b Bus)
113. (related b a inFrontOf)
```

Figure 8: Sample of the ontology in the emergency-vehicle domain.

4. EVENT RECOGNITION

In our approach, the agents are empowered with domain knowledge and they can perform geospatial and temporal reasoning. The domain knowledge is imported as ontologies in KRSS syntax. The geospatial reasoning is performed with AllegroGraph, while event reasoning in RacerPro [9].

4.1 Domain knowledge

To facilitate situation awareness, the vehicular agents import knowledge from several sources such as i) vehicular ontologies and ii) street topology.

Firstly, the agents rely on various *vehicular ontologies*, as the one exemplified in figure 8. The ontology consists of the tbox `EmergencyVehicles` (line 102), which defines the main concepts and relationships among these concepts. The relationship `onSameStreet` has the inverse role `onSameStreet`. Both roles relate individuals of type vehicle with instances of the same type (the domain and range contraints in line 104). Buses and emergency vehicles are subsumed by the more generic concept `Vehicle` (lines 105-107). If a vehicle is identified as a `Bus`, it cannot be interpreted as an `EmergencyVehicle` in future instances of time, constrained by the disjoint property in line 108.

In the abox `emergency-vehicles-salonic`, the ontology contains assertions about a particular situation, in which the bus `b` is in front of the ambulance `a` (line 112). Based on axiom 109, both vehicular agents can deduce that `a` is on the same street with `b`. Note that the domain and range restrictions of the role `onSameStreet` are satisfied, because both `b` and `a` are vehicles.

Secondly, street topology data is obtained from *OpenStreetMap*, which gives to vehicular agents the possibility to choose an area by its coordinates and to export it in the RDF format. The RDF tuples include knowledge about the selected traffic area: the location of semaphores, lanes, one way street, etc., that will be exploited during the reasoning process.

4.2 Geospatial continuous reasoning

We employ reasoning in description logic on top of the Geocast protocol [8] to determine the location and status of the vehicles situated in the area of interest. By complementing the location table and the information from the periodically beaconing messages with domain knowledge, a vehicle is able to infer various information: (i) based on the node ID and the speed entries in the location table, the system can infer if a vehicle is moving or it is stationary; (ii) based on the node ID, the geographical position, speed and heading

```
SELECT ?car ?p {?car ex:location ?p.
       ?car onStreet ex:str1.
} ORDER BY
  <http://franz.com/ns/allegrograph/3.0/geospatial
  /fn/haversineKilometers>
  (?o,POINT(22.939007, 40.640392))
```

Figure 9: SPARQL query for continuously retrieving the vehicles on the same street.

Table 2: Temporal predicates in vehicular streams.

Temporal predicate	Informal semantics
`((move ?a)` t_{start} t_{end}`)`	agent `?a` is known to be moving between time t_{start} and time t_{end}
`((approach ?a1 ?a2)` t_{start} t_{end}`)`	`?a1` is approaching agent `?a2` during the time interval $[t_{start}, t_{end}]$
`((behind ?a1 ?a2)` t_{start} t_{end}`)`	`?a1` is behind agent `?a2` during the time interval $[t_{start}, t_{end}]$
`((beside ?a1 ?a2)` t_{start} t_{end}`)`	`?a1` is beside agent `?a2` during the time interval $[t_{start}, t_{end}]$
`((inFrontOf ?a1 ?a2)` t_{start} t_{end}`)`	`?a1` is beside agent `?a2` during the time interval $[t_{start}, t_{end}]$

```
114. (define-event-assertion ((move c1) 5 60))
115. (define-event-assertion ((move c2) 1 50))
116. (define-event-assertion ((approach c1 c2) 10 20))
117. (define-event-assertion ((behind c1 c2) 10 20))
118. (define-event-assertion ((beside c1 c2) 20 30))
119. (define-event-assertion ((inFrontOf c1 c2) 30 60))
```

Figure 10: Asserting primitive events from vehicular communication about `c1` and `c2`.

```
121. (define-event-rule ((overtake ?i ?j) ?t1 ?t2)
122. ((?i vehicle) ?t0 ?tn)
123. ((?i ?j on-same-line) ?t0 ?tn)
124. ((move ?i) ?t0 ?t2)
125. ((move ?j) ?t1 ?t2)
126. ((approach ?i ?j) ?t1 ?t3)
127. ((behind ?i ?j) ?t1 ?t3)
128. ((beside ?i ?j) ?t3 ?t4)
129. ((in-front-of ?i ?j) ?t4 ?t2)
130. ((on-same-line ?i ?j) ?t4 ?t2))
```

Figure 11: Event recognition: vehicle overtaking.

entries in the location table compared to vehicle data, the system can infer if a vehicle is approaching, if it is in the rear/front or if the ambulance is traveling from the opposite direction; (iii) if the beaconing signal adds lane information, than the system is able to recognize lane-changing events.

The geospatial reasoning capabilities of AllegroGraph are used to retrieve all vehicles on the same street (figure 9). Here, the query also orders the vehicles based on distance to retrieve the closest car. The SPARQL query selects the cars `?car` and positions `?p` on a specific street `str1` and returns an ordered list of pairs `(car, position)` by the distance from the current position. Note the usage of primitive function `haversineKilometers` from AllegroGraph library for converting geospatial locations.

4.3 Temporal reasoning

Stream data received from the vehicular network is used to trigger rules that assert volatile facts about the current situation. Assertions about vehicles are valid only within a certain time interval. The vehicles have to recognize primitive events such as which vehicle is approaching, direction of approaching, moving or stationary vehicles. We use the temporal assertions in table 2 to model the above specifications. The reasoning engine combines contiguous facts in a single temporal assertion. Given two assertions `((behind ?o1 ?o2` t_0 t_k`)` and `((behind ?o1 ?o2)` t_{k+1} t_n`)`, the merged information is stored as `((behind ?o1 ?o2)` t_0 t_n`)`.

Assume the assertions in figure 10 have been stored. The vehicle `c1` is moving between time instances 5 and 60 (line 114), while `c2` between 1 and 50 (line 115). Between timesteps 10 and 20, `c1` is approaching `c2` (line 116). Both vehicular agents are aware that the `c1` is approaching from behind (line 117). Between 20 and 30, the vehicles are beside each other, as identified in line 118. After time 30, `c1` is in front of `c2` (assertion 119). These assertions are obtained from reasoning on the data from vehicular communication.

Rules on top of description logic are enacted to recognize complex events, as car overtaking event in figure 11 (adapted from [9]). The rule identifies if an individual `i` overtakes the individual `j`, and the interval of time `[t1, t2]` required for this maneuver. Primitive volatile facts from table 2 represent premises of the rule. These premises check that: 1) `i` is a vehicle (premise 122); 2) the two cars are on the same lane (premise 123); 3) the vehicles are moving (axioms 124-125); 4) car `i` is approaching for behind (axiom 127); 5) car `i` is for e period of time beside the other vehicle (axiom 128); 6) finally, `c1` gets in front of `c2` (axiom 129) and also on the same lane with the overtaken vehicle (axiom 130). Note also the corresponding time constraints. For instance, the `beside` assertion should start at time `t3`, exactly when `behind` assertion is no longer valid.

5. VEHICLE OVERTAKING SCENARIO

Consider the overtaking with opposing traffic scenario. The vehicle `c` should avoid the leading vehicle `c2` in the originating lane and the leading vehicle `c3` in the opposing lane. Before overtaking, `c1` should check that: i) it is allowed to overtake; ii) `c2` does not signal left; iii) there is sufficient distance to return to the same lane without endangering vehicle `c3` coming from the opposite direction or breaching the norms (i.e. continuous line); iv) no other vehicle is overtaking `c1`, by checking the road behind; v) signal intention to overtake for long enough to warn all other road users. To check if the vehicle `c1` successfully overtook `c2`, the RacerPro query q_1 is used: `(timenet-retrieve ((overtake c1 c2) ?t1 ?t2))`. Given the assertions in figure 12, the system is able to recognize the overtaking event. To retrive all vehicles on a given line, we check the positions belonging to the concepts `Lane1` or `Lane2` as defined by axioms 142-145. The RacerPro query `(concept-instances Lane1)` returns the list `(l1, l2, l3)`. Similarly, the query `(concept-instances Lane2)` identifies the active locations `(l4, l5)`. To simulate different cooperative scenarios in our framework, various types of agents are instantiated (see lines 146-148) and deployed in different topological situations.

6. RELATED WORK

Ontologies and VANETS. Several related ontologies in the automotive domain have been developed [6, 12, 3, 10, 7]. An ontology of vehicular networks security has been modeled in [6] aiming to classify the vulnerabilities based on the impact of the intrusion and functionality affected in

```
(define-event-assertion ((hasLocation c1 l1) 5 6))
(define-event-assertion ((hasLocation c1 l2) 6 7))
(define-event-assertion ((hasLocation c1 l3) 7 8))
(define-event-assertion ((hasLocation c1 l3) 8 9))
(define-event-assertion ((hasLocation c2 l2) 5 6))
(define-event-assertion ((hasLocation c3 l4) 5 6))
(instance l1 (and (= hasLat 40.63935)(= hasLong 22.9446606)))
(instance l2 (and (= hasLat 40.63936)(= hasLong 22.9446606)))
(instance l3 (and (= hasLat 40.63937)(= hasLong 22.9446606)))
(instance l4 (and (= hasLat 40.63938)(= hasLong 22.9446607)))
(instance l5 (and (= hasLat 40.63938)(= hasLong 22.9446607)))
(equiv Lane1 (and (< hasLat 40.63939) (> hasLat 40.63930)
                  (= hasLong 22.9446606)))
(equiv Lane2 (and (< hasLat 40.63939) (> hasLat 40.6393)
                  (= hasLong 22.9446607)))
(instance c1 (and PassiveCooperative NormalAgent Polite))
(instance c2 (and ActiveCooperative Impolite))
(instance c3 (and ActiveCooperative Polite))
```

Figure 12: Assertions in the overtaking scenario.

routing protocols. The ontology in [12] aims to determine the autonomy layer of an automated vehicle, with the main of self-assessment of the perception system to monitor co-driving. The CAOVA (Car Accident lightweight Ontology for VANETs) structures information from two sources: i) collected from vehicle sensors when an accident occurs, or ii) imported from the the General Estimates System accidents database [3]. Due to the private character of data formalized for the driver concept (blood, allergies, illness, medication, pregnant, weight, age, sex), the ontology has been encrypted using the Advanced Encryption Standard. A vehicular ontology is proposed in [10] having the role of a dynamic middleware between two VANETS. The ontology focuses on defining packets and their features (MFRBroadcastPacket, PositionBasedPacket, ClusterBasedPacket, etc). Axioms are used to infer the meaning of a packet and to classify it under the most appropriate routing strategy.

Agents in VANETS. Jade-LEAP is used in [1] to develop vehicular agents communication through mobile devices. In our case, we assume communication through vanet technologies, aiming to be consistent with the IEEE811.p standard. A vanet-based emergency vehicle warning system has been developed in [4]. Four types of meesages have testet for the emergency scenario. Our ontology includes all the message types required by general vanet-related applications.

Overtaking. A system that assists drivers in overtaking maneuvers has been developed in [5] based on a kinematics model and a communication protocol. Similar to our work, the application can detect ongoing overtaking maneuvers. Instead of focusing on an accurate kinematics model, our solution exploits domain knowledge in order to detect events. By interleaving vehicular knowledge with continuous reasoning on vehicular data streams, more high-level events can be defined and identified in case of their occurrence.

7. CONCLUSION

We argue that the integration of agent technologies with network communication has huge potential in the context of vehicular networks. The contributions of this paper are: i) defining a model to encapsulate vehicualr data as assertion in an ontology ii) formalise messages for vehicular communication in description logic iii) formalise agent types for the overtaking scenario. The agents were empowered with temporal and spatial reasoning capabilities in order to reason on volatile vehicular knowledge.

8. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments. This research is supported by Technical University of Cluj-Napoca, Romania through the internal project "GREEN-VANETS: Improving transportation using Car-2-X communication and multi agent systems".

9. REFERENCES

[1] M. Amor, I. Ayala, and L. Fuentes. A4vanet: Context-aware jade-leap agents for vanets. In *Advances in Practical Applications of Agents and Multiagent Systems*, pages 279–284. Springer, 2010.

[2] F. Baader. *The description logic handbook: theory, implementation, and applications.* Cambridge university press, 2003.

[3] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni. Caova: A car accident ontology for vanets. In *Wireless Communications and Networking Conference (WCNC)*, pages 1864–1869. IEEE, 2012.

[4] A. Buchenscheit, F. Schaub, F. Kargl, and M. Weber. A vanet-based emergency vehicle warning system. In *Vehicular Networking Conference (VNC), 2009 IEEE*, pages 1–8. IEEE, 2009.

[5] A. S. de Sousa Vieira, J. Celestino Júnior, A. Patel, and M. Taghavi. Driver assistance system towards overtaking in vehicular ad hoc networks. In *AICT 2013, The Ninth Advanced International Conference on Telecommunications*, pages 100–107, 2013.

[6] M. Erritali, B. El Ouahidi, B. Hssina, B. Boukhalene, and A. Merbouha. An ontology-based intrusion detection for vehicular ad hoc networks. *Journal of Theoretical & Applied Information Technology*, 53(3):410–414, 2013.

[7] M. Feld and C. Müller. The automotive ontology: managing knowledge inside the vehicle and sharing it between cars. In *3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 79–86. ACM, 2011.

[8] A. Festag, R. Baldessari, W. Zhang, L. Le, A. Sarma, and M. Fukukawa. Car-2-x communication for safety and infotainment in europe. *NEC Technical Journal*, 3(1):21–26, 2008.

[9] Racer Systems GmbH & Co. KG Racer user guide 2.0, 2012.

[10] V. Nundloll, P. Grace, and G. S. Blair. The role of ontologies in enabling dynamic interoperability. In *Distributed Applications and Interoperable Systems*, pages 179–193. Springer, 2011.

[11] S. Olariu and M. C. Weigle. *Vehicular networks: from theory to practice.* CRC Press, 2010.

[12] E. Pollard, P. Morignot, and F. Nashashibi. An ontology-based model to determine the automation level of an automated vehicle for co-driving. In *FUSION*, pages 596–603. IEEE, 2013.

[13] J. T. Pollock and R. Hodgson. Ontology design patterns. *Adaptive Information: Improving Business through Semantic Interoperability, Grid Computing, and Enterprise Integration*, pages 145–194.