Advanced Driving Behavior Analytics for an Improved Safety Assessment and Driver Fingerprinting

Afaf Bouhoute, Rachid Oucheikh, Karim Boubouh, and Ismail Berrada

***Abstract*— The recent computerizations of cars, together with the development of sensor technologies and car communication devices have transformed the cars into wealthy sources of information. The analysis of data generated continuously by cars can contribute greatly in improving driving safety and drivers comfort. Even though different analytical solutions have emerged recently, there still exist some important issues in driving safety that we assume were poorly addressed, as well as diverse mathematical methodologies whose application in driving behavior analysis is to be investigated. In this paper, we developed a methodology to process and analyze car-generated data, with focus on two analysis goals: 1) automatic verification of drivers' behavior conformity to traffic rules; and 2) visualization and comparison of drivers' behaviors. The proposed methodology is divided into three steps. At first, the abstraction using numerical domains is used to reduce the size of the generated data. Then, the probabilistic graphical models (Probabilistic Automata, and Labeled Directed Graphs) combined with a machine-learning algorithm are used for building a formal model of the driver behavior. Finally, two indepth analyses are carried out by applying automatic model checking and graph matching techniques. Early experimental results point out that the design of numerical domains considered influences hugely the analysis results.**

***Index Terms*— Driving behavior, driving data analysis, driving safety verification, drivers similarity, model checking, graph-based analysis.**

I. Introduction

EVER since the dawn of the car, driver behavior has been the topic of different studies. Researchers from different backgrounds (psychology, transport engineering, car industry, etc.) have been trying to understand and analyze human driving behavior and even investigating its predictability, in order to reinforce traffic safety and enhance drivers comfort.

The computerization of cars, the development of sensors technologies and in-vehicle networking solutions as well as the introduction of vehicular communication offer today new wealthy sources of information on the driver, the vehicle and the environment. Such information allows to study the driver behavior in realistic situations, and to better understand the relations between the driver actions, the vehicle performance and the driving environment. On the other hand, new generations of cars are generating more and more voluminous data. According to IBM [1], car sensors can produce about 1.3 gigabytes of data every hour, and an estimated 312 million gigabytes yearly for 4 hours of daily driving. Therefore, it is important to emphasize that automotive solutions are facing a big data challenge [2] and more powerful analysis tools are needed.

Driving behavior analytics solutions process vehicle data and transform it into valuable information, with data visualization functionalities to facilitate the presentation of results. We cite as examples of existing commercial solutions, *i4drive Driving Analytics* [3], *CLOUD MADE's Car and Driver Analytics* [4] and *Zendrive* [5]. In addition to in-vehicle data, advanced solutions (such as AMODO driving analytics [6]) use data from external sources to improve their analysis. Considered as value-added functions for ITS (Intelligent Transportation Systems) solutions for smart city projects, more attention is paid to the aforementioned analytics; and governments as well as many companies are investing in this field.

Generally, the process of data analysis involves three common tasks: ***acquisition***, ***pre-processing***, and ***modeling and analysis***.

Driving behavior related data can be acquired usually by on-board devices and/or by road-side sensors. In Vehicle Data Recorders (IVDRs) are one of the tools widely used for on-board data collection. They are devices installed on vehicles that monitor and record continuously the vehicle parameters. Even though the first commercialized applications of IVDRs were merely tracking systems designed for fleet management [7], the concept was later extended and other applications have appeared (e.g. car insurance). Other studies such as in *100-Car naturalistic study* [8] and *international large-scale vehicle corpora* [9] focus on the collection of large amounts of naturalistic driving data, for use in driver behavior research.

Using models to represent the driver behavior is of relevance for driving data analysis. Statistical methods combined with methods from artificial intelligence and machine learning have been widely used in driver behavior studies [10], [11]. Actually, the approach used for driver behavior modeling depends primarily on the goals of the driving analysis

Manuscript received February 16, 2017; revised June 29, 2017 and February 26, 2018; accepted July 30, 2018. This work was supported by the National Center of Scientific and Technical Research, Morocco. The Associate Editor for this paper was X. Ban. *(Corresponding author: Afaf Bouhoute.)*

A. Bouhoute, K. Boubouh, and I. Berrada are with the Laboratory of Informatics, Modeling and Systems, Department of Computer Science, Faculty of Science and Technology of Fez, Sidi Mohamed Ben Abdellah University, Fes-Atlas 30003, Morocco (e-mail: afaf.bouhoute@usmba.ac.ma; karim.boubouh@usmba.ac.ma; ismail.berrada@usmba.ac.ma).

R. Oucheikh is with the Cyber-Physical Systems Laboratory, Norwegian University of Science and Technology, 6025 Ålesund, Norway (e-mail: rachid.oucheikh@usmba.ac.ma).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2018.2864637

system and its applications [12]. While some analyses focus on studying and understanding different facets of drivers behavior in particular situations [13], [14], other applications use driving behavior analysis for **recognition, detection** (of drivers, maneuvers, accidents etc.) and/or **classification** (driving styles, etc.) purposes. Graphical models such as Hidden Markov Models (HMMs) have been applied in several analyses for routes and driving maneuvers recognition [15]–[17]. Whereas classification methods such as Artificial Neural Networks, Support Vector Machines clustering and fuzzy logic based algorithms have been used in applications dealing with pattern recognition and classification of driving behavior styles and drivers states [18]–[20]. Beyond recognition and classification, recent studies in driving behavior analysis are investigating drivers heterogeneity and the possibility of driver fingerprinting. In fact, drivers are distinguishable, even when little data and few sensors are available [21]. Existing approaches that address driver identification, have applied and combined statistical methods (Gaussian Mixture Model) with machine learning algorithms (Neural Networks, SVM, K-means, etc.) [21]–[23]. However, the heterogeneity in driving behavior has been addressed differently in [24], where authors propose Driving Habit Graph (DHG) to represent driver behavior. The DHG models the driving style of one driver in different driving maneuvers, extracted from numerical sensor data. The DHG of a driver, consists of macro nodes, representing the driving maneuvers, linked by arcs indicating the order of their occurrence. The macro nodes are modeled as a Driving Relation Map (DRM), a subgraph built from raw data with nodes reflecting the significant changes in the driving signals.

In this paper, we propose an approach for analyzing driving behavior using automotive sensors data (speed, acceleration, steering angle, etc.). As a very first step, to prepare the raw data for analysis, abstraction using the interval domain is used. This abstraction transforms the raw measurements into a form of intervals, ignoring thus irrelevant details from instantaneous measurements. We propose then two graphical models to represent driving behavior, which are probabilistic hybrid automata and labeled directed graphs. We choose the use of graphical models because of their merging of graph theory and probability theory, which makes them one of the most applied solutions in data representation and interpretation. Finally based on these models, two analyses are performed: in the first analysis, model checking of the automaton model of the driver behavior is used to verify the compliance of his actions with the road rules; while the second uses graph matching theory to allow a comparison of the behavior of different drivers.

The remainder of this paper is organized as follows. Section II addresses the numerical abstraction of sensor data. Section III details the two models used for representing driver's behavior. Section IV presents our approach for conducting a safety analysis of the recorded driving behavior. Section V introduces our method for analyzing the similarity between drivers, its use for drivers identification is then investigated. Finally, section VI concludes the paper.

TABLE I
EXAMPLES OF DRIVING DATA

Variable	Description
Speed	The car speed
Engine speed	The engines RPM (Revolutions Per Minute)
Acceleration	Acceleration of the car
Steering wheel angle	The position angle of the steering wheel
Heading	The vehicle heading
Fuel level	Level of fuel contained in the fuel tank
Brake/accelerator pedal pressure	The status of the brake/accelerator pedal
Clutch pedal position	Position of the clutch pedal
Transmission gear position	State of the transmission gear
Parking brake	Status of the parking brake

II. DRIVING DATA PREPROCESSING

Data acquisition systems, continuously operating, collect a tremendous amount of heterogeneous data through a multiplicity of in-vehicle sensors. Such sensors allow the measurement of information on variables related to the vehicle, the driver and the surrounding environment. Accelerometers and speedometers, for instance, are used to measure acceleration and speed of the car; radar and lidar sensors for distance to obstacles measurement and GPS for location coordinates, etc. We give in Table I examples of variables related to vehicle operation. The multiplicity of variables which can realistically be collected results in a very large quantity of collected data. This large size of initial data may, in fact, yield much richer information but also makes the analysis impractical or infeasible. For a practical and more effective analysis, driving behavior analysis approaches must use some abstraction techniques that will reduce the size of data and facilitate its manipulation and evaluation. This section presents the approach we have used for driving data abstraction.

A. Numerical Abstraction

The concept of abstraction has been used in many different domains from social theory, philosophy to mathematics, and computer science. In theoretical computer science, Patrick and Radhia Cousot have introduced the abstract interpretation theory for abstracting mathematical structures, involved in the formal models of computer systems [25]. The purpose of this theory is to build a sound approximation of the behaviors of complex computer systems, which is used to facilitate the reasoning on and the verification of their behavioral properties (non-termination, correct termination or with errors, etc.). Though it was primarily defined for static program analysis, abstract interpretation has been applied in other problems in computer science (model checking, database queries, malware detection) [26]. One key concept in abstract interpretation is that of abstract domains used for approximation. The choice of an abstract domain is based on the analyzed system structure and the properties to be inferred. Among the most commonly used domains are intervals, pentagons and octagons domains.

B. Abstraction of Driving Data Using Interval Domain

For driving data abstraction, we propose an adaptation of the interval domain. We choose to use the interval domain

because it is more adapted to the models that we will use later. Each collected variable is abstracted separately. For the sake of simplicity, we have ignored any correlation between the driving data (such as between acceleration and speed or pedal pressure and acceleration), so we can deal with non-relational domains.

Let $\mathcal{I}_{\bar{\mathbb{R}}} = \{\bot\} \cup \{[a, b[\mid a \in \mathbb{R} \cup \{-\infty\},\ b \in \mathbb{R} \cup \{+\infty\}\ and\ a < b\}$ the set of right-open intervals with bounds in $\mathbb{R}$ and where $\bot$ is the empty interval. The concrete domain of x is defined as the poset $C_x = \langle \bar{\mathbb{R}}, \subseteq \rangle$ where $\subseteq$ is the usual inclusion operator. Whereas, the abstract domain poset $A_x = \langle \mathcal{I}_{\bar{\mathbb{R}}}, \sqsubseteq \rangle$ defines the abstract domain equipped with the ordering relation $\sqsubseteq$. $\sqsubseteq$ is defined by $[a, b[\sqsubseteq [a', b'[\iff a \geq a'\ and\ b \leq b'$, and we define the two operators $\sqcap$ and $\sqcup$ by the following:

$$X \sqcap Y = \bot \ \textbf{\textit{if}}\ Y = \bot\ \textbf{\textit{or}}\ X = \bot$$

$$X \sqcap Y = \begin{cases} [max(a, a'), min(b, b')[\\ \quad \textbf{\textit{if}}\ X = [a, b[\ \textbf{\textit{and}}\ Y = [a', b'[\\ \quad \textbf{\textit{and}}\ max(a, a') \leq min(b, b') \\ \bot\ \ \textbf{\textit{otherwise}} \end{cases}$$

$$X \sqcup Y = \begin{cases} [min(a, a'), max(b, b')[\\ \quad \textbf{\textit{if}}\ X = [a, b[\ \textbf{\textit{and}}\ Y = [a', b'[\\ X \quad \textbf{\textit{if}}\ Y = \bot \\ Y \quad \textbf{\textit{if}}\ X = \bot \end{cases}$$

In this interval abstract domain, a set c of reals is approximated with the interval $[a, b[$ where $a = min(c)$ and $b = max(c)$. The abstraction function α_x is thus defined as:

$$\begin{aligned} \alpha_x : \bar{\mathbb{R}} &\to \mathcal{I}_{\bar{\mathbb{R}}} \\ c &\mapsto [min(c), max(c)[, \\ \alpha_x(\emptyset) &= \bot \ \ if\ c \neq \emptyset \end{aligned}$$

The concretization function γ_x is conversely defined as:

$$\gamma_x([a, b[) = \{y \in \mathbb{R} \mid a \leq y < b\} \ \ and\ \gamma_x(\bot) = \emptyset.$$

It is easy to see that the functions (α_x, γ_x) form a Galois connection $\langle C_x, \subseteq \rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle A_x, \sqsubseteq \rangle$, and we can say then that A_x is a sound abstraction of C_x.

In order to define the numerical intervals for driving data, we distinguish between two approaches, static approach and dynamic approach.

In the static approach, the intervals are generally defined depending on the scope and the intended goals of the driving data analysis. Let us take for example the variable "vehicle speed" denoted as vel, which values may range from 0 to 240 km/h (the concrete domain). In case of an application that analyzes the compliance of a driver to speed limit panels, vel can be abstracted based on the speed limit panels that exist in the traffic code. The French traffic code defines 6 speed limit panels [27]: 30, 50, 70, 90, 110, 130. vel can therefore be abstracted by intervals from the set $A_{vel} = \{[0, 30[, [30, 50[, [50, 70[, [70, 90[, [90, 110[, [110, 130[, [130, 240[, [240, +\infty)\}$. Explicitly, all values of vehicle speed ranging from 0 to 30 will be represented as one abstract value that is the interval $[0, 30[$ thus ignoring the speed changes between 0 and 30 and recording only changes between the significant values for the considered application (abstract values). Using the abstraction and concretization functions this can be written as $\forall y \in \mathbb{R}$ such that $0 \leq y < 30, \alpha_{vel}(y) = [0, 30[$ while $\gamma_{vel}([0, 30[) = \{y \in \mathbb{R} | 0 \leq y < 30\}$. This abstraction is shown graphically in Fig.1.

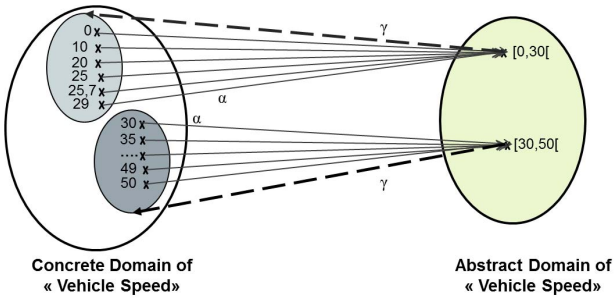

Fig. 1. Abstraction using the interval domain, of the variable "Vehicle Speed" (α is the abstraction function, γ is the concretization function).

Actually, the example above as well as the applications we presented later in this paper use a static abstraction, which means that the intervals of the abstract domain were set in advance. However, a dynamic approach can be considered where machine learning algorithms are used to retrieve the most suitable intervals from available data. An example of the methodology for dynamic data abstraction using the k-means algorithm is presented in the next paragraph.

C. Dynamic Abstraction of Driving Data Using k-Means

k-means is a popular clustering method commonly used to automatically partition a dataset into k groups [28]. Its popularity is presumably due to its speed and its relative simplicity of deployment. Basically, the k-means algorithm takes as input a dataset $X = X_1, X_2, \ldots, X_N$ of N points and an integer k (i.e. the number of clusters to be generated). At the end, the algorithm generates k cluster centroids (i.e. used to define the clusters) and assign each data point to a unique cluster. The clustering is performed such that the distance between all points and the centroid within a cluster is shorter than any other cluster centroid. This condition is generally expressed by the following expression, where C_i, and μ_i are used to denote respectively a cluster i (among the k clusters) and its centroid:

$$\text{Minimize} \sum_{i=1}^{k} \sum_{x \in C_k} ||x - \mu_i||^2 \text{ with respect to } C_i, \mu_i$$

In our context, each cluster will be represented by an interval defined by the min and max values within the cluster. In this way, the numerical abstraction of a driving variable consists of a segmentation of its values into k disjoint intervals, based on the k-means algorithm. The number of intervals k is estimated from the available data. As we presented in the static approach, each variable from the driving dataset is segmented separately. For more significant reduction of data, principal component analysis (PCA) is carried out to

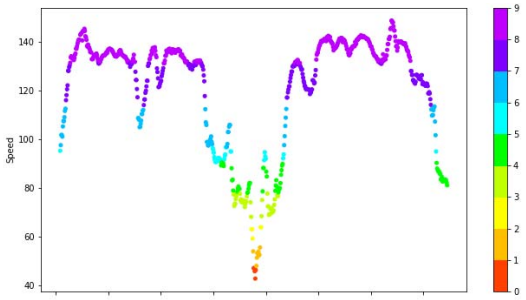

Fig. 2. Dynamic abstraction of speed by means of the k-means algorithm. The number of clusters generated is $k = 9$.

select the most relevant and uncorrelated variables. Fig.2 shows an example of segmentation of the variable speed measured for a driver D1 over a trip T, taken from the public dataset UAH-DriveSet[1] [29]. The application of k-means results in the generation of 9 clusters of the speed values. In different colors are illustrated the different intervals to be used for speed abstraction. Thus, the variable speed can be abstracted by intervals from the set $A_{speed} = \{[0, 47.1[, [47.1, 56.2[, [56.2, 65.7[, [65.7, 78.0[, [78.0, 90.6[, [90.6, 97.0[, [97.0, 113.7[, [113.7, 131.3[, [131.3, 148.8[, [148.8, 255[, [255, \infty)\}$.

An other interesting idea may be to apply the k-means clustering on all driving variables instead of a separate clustering. This is expected to reduce significantly the size of the data.

To summarize, we can say that the dynamic approach for abstraction is data-adapted, which allow to create abstract domains that fit better to each dataset. In the other hand, the static approach is more adapted in specific applications, *i.e. such as analyzing the compliance of the driver with traffic code (as mentioned in the example in the previous paragraph), by fixing the intervals manually it will be easier to verify whether the driver is in authorized/unauthorized state*, and when dealing with different datasets to create a uniform space of the abstract states.

III. Data-Based Modeling of Driving Behavior

Driving data contains a lot of information about the behavior of the driver, and its interactions with the vehicle and the environment. However, to extract a maximum of useful information from the data, a modeling approach is needed. A formal model created from data will help to understand the behavior of the driver and facilitate its analysis. In the previous section, we have introduced a numerical abstraction of the driving data using the interval domain. In the following, we present two data-based models to represent the driver behavior. The first model is based on automata modeling formalism, in which the driving behavior is represented as an automaton operating in the driving environment. The use of automata-based modeling for driver behavior representation makes it possible to use model checking tools for formal verification of human driving behavior. The second modeling approach proposes a graph based representation of the driver behavior, which will allow us to use graph matching algorithms to mathematically compare drivers behaviors.

[1]available at http://www.robesafe.uah.es/personal/eduardo.romera/uah-driveset/

A. Driving Behavior as a Probabilistic Rectangular Hybrid Input/Output Automaton (PRHIOA)

Probabilistic Rectangular Hybrid I/O Automaton was presented in [30] as a framework for the modeling of human driving behavior that captures all the interactions of the driver with its vehicle and the environment. It is based on the formalism of rectangular automata for which the reachability problem is proved to be decidable [31]. Readers can refer to [30] for the formal definition of PRHIOA.

An automaton representation of driving behavior consists of the following:

- ***Internal variables***. The set of driving data related to the state of the vehicle such as *speed*, *acceleration*, *steering angle*, etc. To keep contextual information about the driving environment in each state (traffic laws and driving rules mainly), a variable which we refer to as the driving context *cxt* is considered. This latter is a representation of the driving requirements as a set of conditions on the driving variables, and which is updated continuously due to the dynamic nature of the driving environment [30].
- ***States***. The set of states correspond to the possible valuations of driving data.
- ***Invariants***. A state invariant is defined as a conjunction of conditions over the internal variables. A condition over a variable x takes the form of an interval $[a, b[$ from the abstract domain of x.
- ***Input actions***. The representation of contextual information about the driving environment. They consist of real time information about traffic situations received either through in-vehicle sensors or through vehicular communication device. Examples of inputs include prescription road signs (like speed limitation), and warnings of coming vehicles. The inputs are used to construct and update the driving context *cxt*. The update follows predetermined rules that were designed depending on the nature of the different inputs.
- ***Output actions***. A modeling of driver response to driving environment requirements. They represent the driver's actions related to driving tasks. For instance, accelerating, turning and stopping are modeled as outputs in the automaton model.
- ***Transitions***. Two kinds of transitions are distinguished: input-enabled transitions that occur whenever an input is received from the environment and output transitions enabled by the driver operation without an explicit input.

The automaton representing the driver behavior is either constructed offline using recorded driving data, or online when sensors measurement are still streaming. The number of the automaton states depends on the cardinalities of the abstract domains defined for the different variables, elements of which are used for the definition of invariants. In fact, the construction is a machine learning algorithm, based on the learning automata algorithm [32], that learns and updates

Require: Q: states, I: input action set, O: output action set, $q_{initial}$: initial state, $Data$: driving data set
Ensure: D: transition set, P: transition probabilities
  $q_{current} = q_{previous} = q_{initial}$
  while Data **do**
    convert $H \in Data$ to its corresponding abstract value $\alpha(H)$
    pick a state q from Q such that $inv(q) = \alpha(H)$
    $q_{current} = q$
    if (input action $I_j \in I$ exists in $Data$) **then**
      $q_{previous} = q_{current}$
      Update current driving context cxt in $q_{current}$: $q_{current}.cxt = cxt_j$
      if $(q_{previous}, q_{current}, I_j) \notin D$ **then**
        Add input-enabled transition T to D
      end if
      Update transitions probabilities using (1) and (2)
    end if
    if (output action $O_j \in O$ exists in $Data$) **then**
      if $(q_{previous}, q_{current}, O_j) \notin D$ **then**
        Add output transition T to D
      end if
      $q_{previous} = q_{current}$
      Update transitions probabilities using (1) and (2)
    end if
  end while

Fig. 3. The algorithm of the automaton-based model construction.

the probabilities of the transitions between states according to the interactions of the driver with the environment. The pseudo code of the algorithm is depicted in Fig.3. The algorithm runs over the available driving data: transitions are added and their probabilities are updated whenever input or output actions occur, using the following reinforcement scheme:

$$P(T) = \begin{cases} 1 & \text{if } r = 0 \\ P(T) + \dfrac{1 - P(T)}{r} & \text{if } r \neq 0 \end{cases} \quad (1)$$

$$P(T') = P(T') - \frac{P(T')}{r}, \quad for\ all\ T' \neq T \quad (2)$$

Where $P(T)$ is the probability of the reinforced transition $T \in D$ outgoing from state q, and r is the number of all transitions T' outgoing from q.

Fig.4 illustrates an example of driving behavior automaton. The variables velocity, denoted vel, and steering wheel angle, denoted θ, are considered as internal variable and we add the variable cxt representing the driving context. As stated before, the expression of cxt is extracted from driving rules imposed by the traffic code in each state. In fact, regulations defined by traffic laws are either implicit (general requirement such as minimum and maximum speed limits on highways, prohibition of u-turn and parking in tunnels) or explicit (road signs, indications, etc.). Initially, with no explicit regulation cxt is initialized to C_0, the implicit requirement of the driving environment. Actually, the implicit requirement depends primarily on road types: urban areas in France, for instance, require a

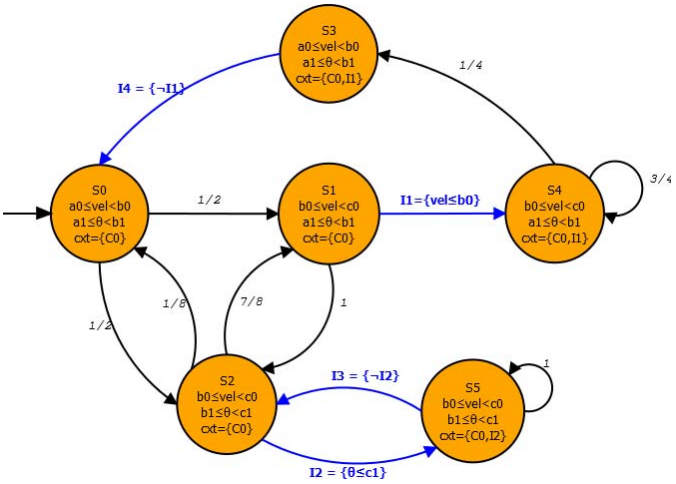

Fig. 4. Example of automata-based representation of driving behavior.

speed limit of 50km/h, another general rule is the obligation of driving on the right side of two-lane roads. The arrows of the automaton labeled with numbers represents transitions initiated by driver actions where the numbers represent the probabilities of their occurrence computed during the learning process. Whereas the other arrows represent input-enabled transitions initiated by the environment inputs and which affect the context cxt.

The automaton depicts some usual behaviors of the driver: for instance, we can say that usually in state S_0, the driver either increases the car velocity or both the velocity and the steering angle, each with a probability of $1/2$. Also in state S_4, the probability of the driver's compliance to the restriction on velocity ($vel \leq b0$) is $1/4$, because in most of the times the driver does not decrease the velocity below $b0$, its values stay in the interval $[b0, c0[$ with a probability of $3/4$. Another behavior is when restriction on θ (e.g when turning or overtaking is prohibited) ended, the driver often (probability of $7/8$) change both the velocity and the steering wheel.

B. Driving Behavior as an Attributed Directed Graph

The second contribution in this section consists of using graphs as formal models for driving behavior modeling. In fact, graphs have been used in many areas of computer science [33] such as software engineering, data mining and networking. Depending on the context of use, we find different types of graphs with various definitions. In this paper, we propose the use of *attributed directed graphs* to represent the behavior of drivers. An *attributed directed graph* G [34], [35] is formally defined as a tuple (V, E, μ, δ) where:

- V is a finite set of vertices.
- $E \subseteq V \times V$ a set of directed edges. An edge $e = (s, d)$ is an arrow defined by the pair from the source vertex $s \in V$ to $d \in V$ the destination vertex.
- $\mu : V \rightarrow L_V$ is a vertex labeling function that associates to each vertex from V labels from the set L_V.
- $\delta : E \rightarrow L_E$ is an edge labeling function that associates to each edge from E labels from L_E.

Require: V: A set of labeled vertices, V_0: initial state, $Data$: driving data set.
Ensure: G: A graph
  $V_i = V_0$, $action$ = *'none'*
  while $Data$ **do**
    map $H \in Data$ to the corresponding label $lbl = \mu(H)$
    while $lbl = \mu(V_i)$ **do**
      wait
    end while
    Pick a vertex V_j with $\mu(V_j) = lbl$
    create an edge $e = (V_i, V_j)$
    if driver action a_i known **then**
      add a_i to the label of e: $\delta(e).action = a_i$
    end if
    if $e \in E$ **then**
      $\delta(e).weight = \delta(e).weight + 1$
    else
      add edge e to E
      $\delta(e).weight = 1$
    end if
    $V_i = V_j$
  end while

Fig. 5. Algorithm of graph construction.

L_V and L_E represent respectively the sets of vertex-labels and edge-labels, where labels are tuples of fixed-size.

Similarly to the automata-based modeling approach, a graph of driving behavior is defined such that:

- *The vertices* represent the dynamic states of the vehicle.
- *The edges* represent the transitions between the states of the vehicle.
- *The vertex-labeling function* assigns to each vertex v a label l_v. A vertex label is a combination of labels assigned to the abstract values of driving data. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of driving data, and L_{x_i} be the set of labels of x_i. We define the labeling function ϵ_{x_i} for variable x_i as:

$$\begin{array}{rccc} \epsilon_{x_i} : & I_{\tilde{\mathcal{R}}} & \rightarrow & L_{x_i} \\ & [a_i, b_i[& \mapsto & l_{i,x_i} \end{array}$$

  The vertex-labeling function μ is defined as:

$$\begin{array}{rccc} \mu : V & \rightarrow & L_V = L_{x_1} \times \ldots \times L_{x_i} \times \ldots \times L_{x_n} \\ v & \mapsto & l_v \end{array}$$

- *The edge-labeling function* assigns to each e a couple $(action, weight)$, where $action$ refers to a driver action and $weight \in \mathbb{R}_+^*$ represents the occurrence of $action$. The edge-labeling function is defined as:

$$\begin{array}{rccc} \gamma : & E & \rightarrow & L_E \\ & e = (s, d) & \mapsto & l_e = (action, weight) \end{array}$$

The algorithm of the graph construction runs over driving data, its pseudo code is presented in Fig.5. At each timestamp, the data read is mapped to a label from L_v, edges are added and weights are updated. The resulting graph will thus be a personalized representation of the driver behavior.

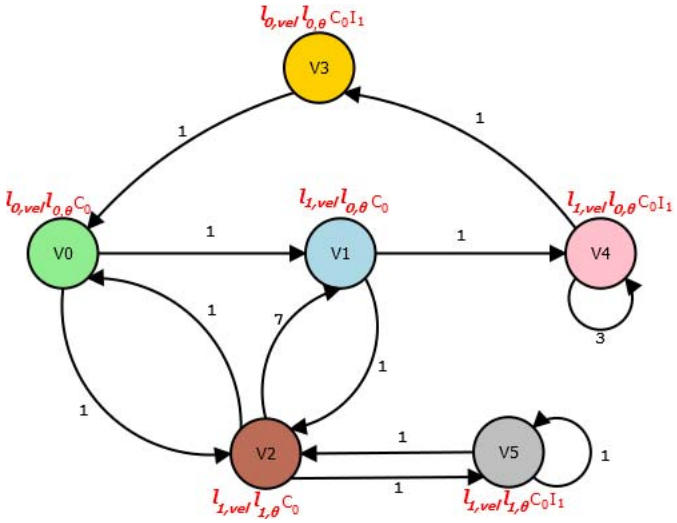

Fig. 6. Example of graph-based representation of driving behavior.

Fig.6 depicts an example of driving behavior graph, representing the same driver of Fig.4. Velocity (vel) and steering wheel angle (θ) were considered for the representation of the vehicle states, and their values were used to label the vertices. Considering A_{vel} and A_θ, the abstract domains of vel and θ defined as described in the previous section. Defining labels of vel consists of assigning to each interval $[a_i, b_i[$ in A_{vel} an integer or string label $l_{i,vel}$. The labels $l_{0,vel}$ and $l_{0,\theta}$ for instance refer to the labels assigned to the first interval in A_{vel} and A_θ respectively. For driving context, the label is defined using the notations assigned to the received inputs. C_0I_1 in the label of V_3 means that the driving environment consists of the implicit requirement of the road and the input I_1. In comparison with the automaton in Fig.4, the labels of the graph vertices replace the invariants of states, while the edge weights replace the probabilities of transitions. Driver actions, if known, are added to the labels of the edges.

C. Relationship Between Driving Data Abstraction and the Application Performance

An important issue relevant to performance that should be considered is the relationship between the model size (number of states) and the defined abstraction. This was carried out by conducting an experimentation using sample data from the signal corpus collected by *the behavior signal processing laboratory of Nagoya University* [36]. The corpus uses different kind of sensors to capture real data relevant to the driver, the vehicle and the environment:

- Driver: Driver speech is recorded using microphones, biological signals such as *heart rate*, *skin potential* and *perspiration* are also recorded.
- Vehicle: Recorded driving signals consist of *brake and gas pedal pressure*, *steering wheel angle*, *vehicle speed*, *following distance* and *3D accelerations*. The sampling frequency is 16kHz.
- Environment: Cameras are used to capture the road ahead, GPS positions.

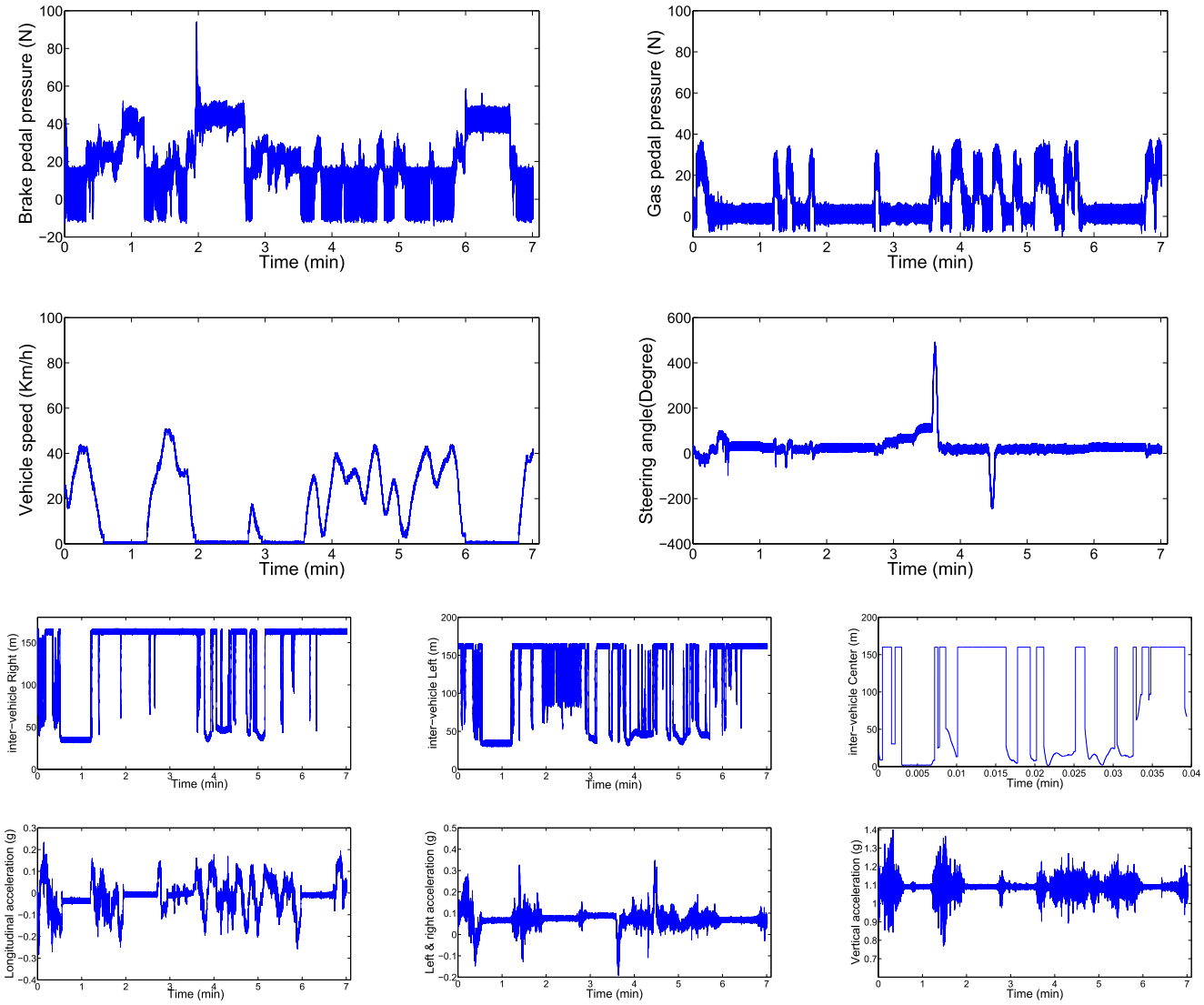

Fig. 7. Distributions of driving signals of one female driver for a 7 minutes drive. The sampling frequency is 16 kHz. The data is provided by CIAIR, Nagoya, Japan (refer to [9] for details).

Fig.7 depicts the measured values of driving signals of one driver (female) during one trip (7 minutes). Numerical abstraction of driving signals can be intuitively seen as a discretization of the signals. We define 5 different abstractions, varying each time the precision with which we abstract the data. The precision is represented by the length of the intervals used in the abstraction. Table II presents the intervals considered for the five abstractions defined for the driving signals used in the experiment. The intervals bounds were defined manually by analyzing the different distributions of data. The lengths of intervals vary according to which signal variations the abstraction should capture.

The model state space is generated for each abstraction separately. Fig.8 compares the size of the model generated for 1-minute drive (model size), using the different abstractions. It shows also the relation between the model size and the time for its generation (processing time). As expected, an abstraction that is very precise (defined using narrow intervals) increases the size of the model and the processing time. Indeed, abstraction 'A1', the closest abstraction to real measurements has the largest model size (6635520 states) and

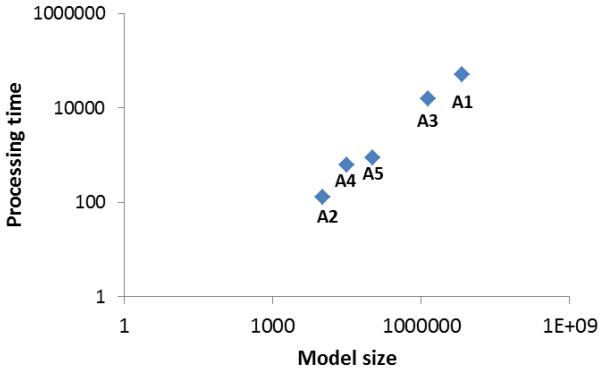

Fig. 8. Relationships between abstraction and performance.

TABLE II
DRIVING DATA ABSTRACTIONS

Driving Signal		Abstractions				
		Abstraction 1 (A1)	Abstraction 2 (A2)	Abstraction 3 (A3)	Abstraction 4 (A4)	Abstraction 5 (A5)
Brake Pedal Pressure (BP)		$\{[-20,0[,[0,20[,[20,40[,[40,60[,[60,80[,[80,100[,[100,\infty[\}$	$\{[-20,20[,[20,60[,[60,100[,[100,\infty[\}$	$\{[-20,0[,[0,20[,[20,40[,[40,100[,[100,\infty[\}$	$\{[-20,0[,[0,40[,[40,100[,[100,\infty[\}$	$\{[-20,20[,[20,50[,[50,100[,[100,\infty[\}$
Gas Pedal Pressure (GP)		$\{[-10,0[,[0,10[,[10,20[,[20,30[,[30,40[,[40,\infty[\}$	$\{[-10,10[,[10,30[,[30,50[,[50,\infty[\}$	$\{[-10,0[,[0,10[,[10,20[,[20,30[,[30,40[,[40,\infty[\}$	$\{[-10,0[,[0,20[,[20,40[,[40,\infty[\}$	$\{[-10,10[,[10,30[,[30,40[,[40,\infty[\}$
Vehicle speed (Vel)		$\{[0,20[,[20,40[,[40,60[,[60,\infty[\}$	$\{[0,30[,[30,60[,[60,\infty[\}$	$\{[0,20[,[20,40[,[40,60[,[60,\infty[\}$	$\{[0,30[,[30,60[,[60,\infty[\}$	$\{[0,30[,[30,60[[60,\infty[\}$
Steering wheel angle (θ)		$\{[-300,-100[,[-100,100[,[100,300[,[300,500[,[500,\infty[\}$	$\{[-300,100[,[100,500[,[500,\infty[\}$	$\{[-300,-100[,[-100,100[,[100,300[,[300,500[,[500,\infty[\}$	$\{[-300,-100[,[-100,100[,[100,500[,[500,\infty[\}$	$\{[-300,-50[,[-50,50[,[50,500[,[500,\infty[\}$
Following distance	*Right front(d_{rf})*	$\{[0,50[,[50,100[,[100,150[,[150,200[,[200,\infty[\}$	$\{[0,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,170[,[170,\infty[\}$	$\{[20,40[,[40,160[,[160,200[,[200,\infty[\}$
	Left front (d_{lf})	$\{[0,50[,[50,100[,[100,150[,[150,200[,[200,\infty[\}$	$\{[0,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,170[,[170,\infty[\}$	$\{[20,40[,[40,160[,[160,200[,[200,\infty[\}$
	Center (d_c)	$\{[0,50[,[50,100[,[100,150[,[150,200[,[200,\infty[\}$	$\{[0,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,170[,[170,\infty[\}$	$\{[20,40[,[40,160[,[160,200[,[200,\infty[\}$
Acceleration	*Longitudinal (a_l)*	$\{[-0.3,-0.2[,[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.1[,[-0.1,0.1[,[0.1,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.2[,[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.2[,[-0.2,0[,[0,0.2[,[0.2,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.05[,[-0.05,0[,[0,0.2[,[0.2,0.3[,[0.3,\infty[\}$
	Directional (a_d)	$\{[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,0.4[,[0.4,\infty[\}$	$\{[-0.2,0[,[0,0.2[,[0.2,0.4[,[0.4,\infty[\}$	$\{[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,0.4[,[0.4,\infty[\}$	$\{[-0.2,0[,[0,0.2[,[0.2,0.4[,[0.4,\infty[\}$	$\{[-0.2,0[,[0,0.1[,[0.1,0.4[,[0.4,\infty[\}$
	Vertical (a_v)	$\{[0.7,0.9[,[0.9,1.1[,[1.1,1.3[,[1.3,1.5[,[1.5,\infty[\}$	$\{[0.7,1.1[,[1.1,1.4[,[1.4,\infty[\}$	$\{[0.7,1.0[,[1.0,1.3[,[1.3,1.5[,[1.5,\infty[\}$	$\{[0.7,1.0[,[1.0,1.3[,[1.3,1.5[,[1.5,\infty[\}$	$\{[0.7,1.09[,[1.09,1.13[,[1.13,1.5[,[1.5,\infty[\}$

thus a longer processing time. Whereas the abstraction with lower precision (defined using wider intervals) ('A2'), has smaller model size and shorter processing time. Therefore, in order to make an abstraction better in terms of performance, a compromise between precision and processing time should be found.

IV. A SAFETY ANALYSIS OF THE DRIVING BEHAVIOR

Driver compliance with road rules, such as speed limits, prohibitions and right of way, is one of the key elements in traffic safety. The assessment of driver behavior conformity to these rules is important in order to distinguish good and bad drivers. To clarify, the behavior of good drivers complies most often with the road rules and requirements in every driving situation, contrast to bad drivers who tends to ignore most of the rules requires by the Highway Code. This ability to assess drivers behavior can be of a great relevance either for traffic controllers, insurance companies, driving schools or for drivers themselves. In this section, we present an approach to enable the automatic checking of whether the driver performs some defined behaviors. For this purpose, we consider Probabilistic Rectangular Hybrid I/O Automaton as driving behavior model and we use model checking to formally verify that it satisfies the properties representing the desired behaviors.

A. Model Checking

Model checking is a formal model-based approach for analysis and testing of systems behaviors. Verification by *model checking* is the act of proving exhaustively, given a model M of a system whether this model satisfies a given specification expressed by logical formula ϕ ($M \models \phi$). Fig.9 illustrates the application of model checking for driving behavior verification. We mean by behavior specification, the expression of the driving behaviors that the driver should satisfy as logical properties. These properties will be fed to the model checker (the model checking tool), which in turn starts a model-checking algorithm for the verification of the

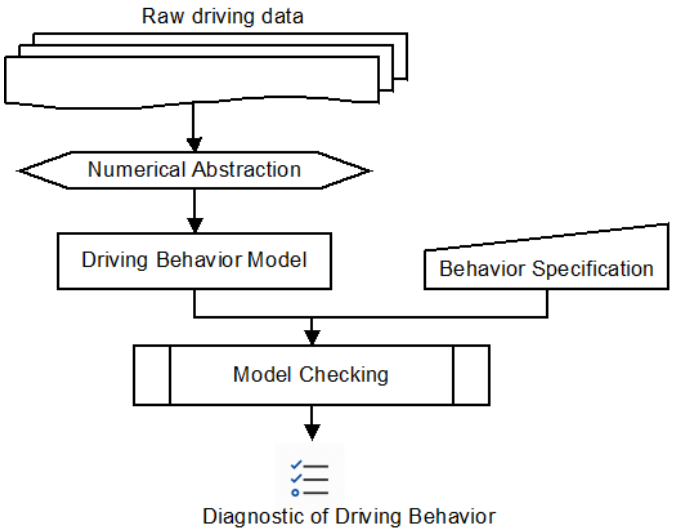

Fig. 9. Driving behavior verification by model checking.

driver behavior model. Generally, there are two main classes of logical properties in formal verification:

- Safety properties: assert that nothing bad happens in the system (e.g. no accident, the driver never transgresses the traffic rules).
- Liveness properties: assert that something good keeps happening (e.g. in all situations the driver can perform an action).

The properties used by the model checker are usually expressed using a formal logic. There exist several logics for expressing properties among which we chose the use of the Probabilistic Computational Tree Logic (PCTL). PCTL is one of the most commonly used logics. It is an extension of the Computational Tree Logic (CTL) allowing probabilistic quantification of properties (e.g. can the driver reach a safe (or unsafe) state with a probability less than 30%). Moreover, PCTL is a branching-time logic considering time as a tree-like structure, which means that properties will be checked

TABLE III
A LIST OF LOGICAL PROPERTIES USED IN VERIFICATION

		Property ϕ	Properties parameters
Non parameterized	$\mathbf{P_1}$	$P =?[F^{<100} d_{c_{min}} > 5/9 \times vel_{max}]$	
	$\mathbf{P_2}$	$A[G\ d_{c_{min}} > 5/9 \times vel_{max}]$	
Parameterized	$\mathbf{P_3}$	$A[G\ d_{lf_{min}} \leq d_{safe} \rightarrow ((\theta_{max} \leq \theta_r\ \&\ \theta_{min} > 0) \mid (\theta_{max} < 0\ \&\ \theta_{min} \geq \theta_l))]$	d_{safe}: the distance considered as safe.
	$\mathbf{P_4}$	$A[G\ d_{rf_{max}} \leq d_{safe} \rightarrow ((\theta_{max} \leq \theta_r\ \&\ \theta_{min} > 0) \mid (\theta_{max} < 0\ \&\ \theta_{min} \geq \theta_l))]$	θ_r: the right turning angle.
	$\mathbf{P_5}$	$A[\ G(a_{l_{min}} \geq a \rightarrow GP_{min} \geq 0)]$	θ_l: the left turning angle.
	$\mathbf{P_6}$	$A[G\ !(GP_{min} > p\ \&\ BP_{min} > p)]$	a: the value of acceleration to not be exceeded
	$\mathbf{P_8}$	$A[\ G(a_{l_{min}} \geq a \rightarrow (\theta_{max} < \theta_r\ \&\ \theta_{min} > \theta_l))]$	p: the tolerable value of pressure
	$\mathbf{P_9}$	$P =\ ?[a_{l_{min}} > 0\ \&\ (\theta_{min} > \theta_r \mid \theta_{max} < \theta_l)]]$	

on all possible paths from a given state. These two aforementioned characteristics, namely the expression of quantitative properties and the tree-like structure, have made of PCTL a suitable choice for our analysis. PCTL defines two categories of formulas: state formulas (Φ) evaluated over a state, and path formulas (φ) evaluated over a path. The syntax of PCTL formulas is defined as follows:

$$\Phi := True|p|\neg\Phi|\Phi \wedge \Phi|\Phi \vee \Phi|P_{\sim\lambda}[\Phi]$$
$$\varphi := X\Phi|P_{\sim\lambda}(\Phi\, \mathcal{U}^{\leq n}\, \Phi)$$

Where p is an atomic proposition, $\sim \in \{<, >, \leq, \geq\}$ is a comparison operator, $\lambda \in [0, 1]$ is a probability threshold and $n \in \mathbf{N} \cup \{\infty\}$.

B. Model Checking of Driving Behavior

In the presence of numerous driving variables, different behaviors can be expressed as logical properties. With no consideration of the driving environment in this analysis (video processing techniques are needed to extract road information from the dataset used in this analysis), only verification of some driving habits such as tailgating, turning behavior, will be considered regardless of the traffic signs. In order to use PRISM model checker [37], the prism model of the driver behavior will contain two local variables for each variable from the initial automaton model, representing the minimum and maximum values of intervals. For instance, vel_{min} and vel_{max} are used to represent the intervals taken by the variable vel. Consequently, in properties expression vel_{max}, resp. vel_{min}, will be used verify whether velocity is lower, resp. greater, than a certain value. For behavior as properties, we distinguish two kinds of properties, *parameterized* properties for which the values used for verification are supplied at verification time and *non-parameterized* properties in which the values are fixed by road regulations. Table III presents a list of properties that we will use in this analysis expressed in PCTL. The two first properties P_1 and P_2 verifying the tailgating behavior are *non-parameterized* properties, because the safe following distance is regulated by the traffic law. The other properties in Table III are *parameterized*. The following driving behavior were considered in this analysis.

- ***Tailgating***. In order to avoid eventual collisions, a driver should keep a safe distance behind its preceding vehicle. For instance, the French Highway Code define the safe following distance (in meter) as $\frac{5}{9} \times vel$, with vel expressed in km/h. This is evaluated by using the properties P_1 and P_2. P_2 will be true if the safe following distance is always respected by the driver, whereas P_1 computes the probability of compliance with the safe following distance rule.
- ***Braking/accelerating behavior***. One dangerous behavior that can damage the car mechanical system is pressing the accelerator and brake pedal at the same time. The driver should not apply pressure on both pedals at the same time. This behavior is expressed by P_6, which will be true if in the model there exists no state with BP and GP both greater than 10. Also, it can be useful to verify the response of the car to the driver action. For instance, we can verify that the car acceleration is positive when the driver presses the gas pedal. This is expressed by The property P_5, which will be true if the model does not contain a state where the longitudinal acceleration a_l is greater than a value a and the gas pedal pressure GP is negative.
- ***Behavior when turning***. A cautious driver should keep his foot off of the gas pedal while turning to avoid a skid. This behavior is expressed by P_8 and P_9. It is also important to avoid crash into obstacles in both sides of the car especially when turning. We propose P_3 and P_4 to express that the driver should keep a safe right front/left front distance from right/left obstacles, which can either be another vehicle or the road side.

C. Discussion

The model checking is carried out by PRISM model checker. The properties, in Table III, have been checked for the models generated by 4 of the abstractions in Table II. Table IV shows the results of properties checking on the model generated for 1, 3 and 7 minutes drive.

To check the veracity of the results, we have compared the frequencies of the different variables. Fig. 10 illustrates the safe following distance calculated from speed values (the red graph) compared with the distance values from the sensor. We notice from the figure that in some points the safe distance is not respected, i.e where the distance is lower than 20m. In fact, checking this behavior by P_2 gives "false" as a result meaning that the subject driver has broken this rule at least once. P_1 gives us more insights because it quantifies the verified behavior. The results show that for 99% of times the

TABLE IV
VERIFICATION RESULTS

Abstraction	A2			A3			A4			A5			Parameters
Recording time(min)	1	3	7	1	3	7	1	3	7	1	3	7	
P_1	0.999949	0.999949	0.999949	0.999999	0.999999	0.999999	0.999652	0.999652	0.999652	0.999793	0.999793	0.999793	
P_2	False	False	False	False	False	False	False	False	False	False	False	False	
P_3	False	False	False	False	False	False	False	False	False	False	False	False	$d_{safe} = 30$,
P_4	True	True	True	True	True	True	True	True	True	True	True	True	$\theta_r = 30$,
P_5	False	False	False	True	False	False	True	True	True	True	True	True	$\theta_l = -30$
P_6	True	True	True	True	True	True	True	True	True	True	True	True	$a = 1$
P_8	True	True	True	False	False	False	False	False	False	True	True	True	$p = 10$
P_9	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$a = 2, \theta_r = 60$, $\theta_l = -60$

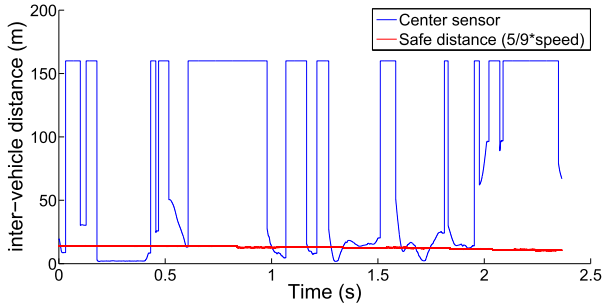

Fig. 10. Following distance reported by the sensor versus safe distance deduced from the speed values.

driver has kept a safe following distance. In the frequency graphs displayed in Fig. 7, we can see that in some points of time where left front distance was ≤ 30, it happens that the steering wheel angle values exceed 30 or are below -30. This was captured by property P_3, proving that the driver does not always keep safe distance from both sides of the car when performing turning maneuvers. P_4 aimed to verify the same behavior considering the right front distance, however, to illustrate the impact of using the min and max variable on the results, we use rfd_{max} in the property expression. As predicted, the checking gives wrong results because by using the upper value, the distance values lower than rfd_{max} are overestimated.

When we compare the verifications for the 4 abstractions, we notice contradictory results for P_5 and P_8. This is actually due to the values of parameters used for verification. For instance, the fact that only abstractions A_2 and A_3 have succeed in capturing a state with longitudinal acceleration greater than $a = 0.1$ and a positive gas pedal pressure is because of the value of $a = 0.1$ is an interval bound which means that the value of interest (0.1) has not been overestimated by the model. We notice also that, in A_3, P_5 becomes false when the recording time increases. However, it is logical because as we record more data, new model states are traversed, and hence states where a property is not satisfied can be found.

In summary, we can say that model checking driving behavior offers a formal approach for an easy traffic safety verification. The results obtained so far have reflected the real behavior of the driver, which was initially represented by frequency graphs of driving factors. Moreover, the use of PCTL logic allowed us to measure the probability of occurrence of verified behaviors. The comparison of results reveals that the verification accuracy depends primarily on the abstraction used and on properties parametrization consequently. Therefore, in order to get correct verification results, the abstraction has to be defined considering the parameters of properties.

V. DRIVER IDENTIFICATION BASED ON DRIVING BEHAVIOR

The driving behaviors in different road conditions vary from one driver to another. For instance, drivers differ in the way they maintain certain speed, how they press the pedals, and also in the distance they keep from other vehicles [38]. Either for personal preferences, habits or vehicle characteristics, drivers tend to have each a unique driving behavior. An analysis that focus on and quantify this heterogeneity is of great applicability for driver identification. The analysis proposed in this section relies on the application of graph theory to represent and analyze driving behavior; since graphs will provide a better view of the driving behavior. Moreover, it will give us the ability to perform graph-based analysis, which was proven useful when applied in many different areas. For instance, we can use graph matching techniques to compare drivers behaviors.

A. Driving Behavior Visualization

With the huge amount of data generated by car sensors, data visualization has become an important means for data interpretation. It consists of using visual graphics to visually communicate information, aiming for a better understanding and analysis of complex data. While frequency distribution graphs are used to analyze driving behavior patterns over time, the graph representation, as proposed in section 3.2, allows an abstract representation and makes the visual comparison easier.

As mentioned earlier in this paper, several researches have focused on the collection and measurement of vehicle data. Ford Bug Labs, for instance, have developed the *OpenXC platform*, an open source API for accessing vehicle data, and made it available for research and automobile applications development. Also, *OpenXC* provides developers with vehicle trace files of anonymous drivers, collected in different scenarios, and for different behaviors. *OpenXC* vehicle trace files are downloaded from *OpenXC* website [39]. The files stated in Table V were used in the present analysis.

TABLE V

OpenXC VEHICLE TRACE FILES USED IN THE ANALYSIS

Vehicle Trace Files for New York City, USA	
Downtown, Crosstown	*East Downtown*
West Downtown	*West Downtown 2*
Uptown Crosstown	*Uptown Crosstown 2*
West Uptown	*West Uptown 2*

TABLE VI

OpenXC DRIVING SIGNALS

Signal	**Range**	**Frequency**	**Abstract Domain Used**
engine_speed	0 to 16382 RPM	max 10Hz	{[0,500[, [500,1000[, [1000,1500[, [1500,2000[, [2000,2500[, [2500,3000[, [3000,4000[, [4000,16382[, [16382,∞)}
fuel_level	0-100%	max 2Hz	{[0,20[, [20,40[, [40,60[, [60,80[, [80,100[, [100,∞)}
torque_at_transmission	-500 to 1500 Nm	max 10Hz	{[-500,-100[, [-100,0[, [0,50[, [50,100[, [100,150[, [150,200[, [200,500[, [500,1500[, [1500,∞[}
transmission_gear_position	{first, second, third, fourth, fifth, sixth, seventh, eighth, reverse, neutral}	1Hz	N/A*
vehicle_speed	0 to 655 km/h	max 10Hz	{[0,10[, [10,20[, [20,30[, [30,40[, [40,50[, [50,60[, [60,655[,[655,∞[}

* N/A: Non abstraction applicable.

TABLE VII

MATCHING THE VERTICES LABELS TO ABSTRACT VALUES

label	**fuel_level**	**engine_speed**	**torque_at_transmission**	**vehicle_speed**	**transmission_gear_position**
-1	N/A**	N/A	"reverse"	N/A	N/A
0	NO_DATA*	NO_DATA	NO_DATA	NO_DATA	"neutral"
1	[0, 20[	[0, 500[	[-500, -100[	[0, 10[	"first"
2	[20, 40[	[500, 1000[	[-100, 0[	[10, 20[	"second"
3	[40, 60[	[1000, 1500[	[0, 50[	[20, 30[	"third"
4	[60, 80[	[1500, 2000[	[50, 100[	[30, 40[	"fourth"
5	[80, 100[	[2000, 2500[	[100, 150[	[40, 50[	"fifth"
6	[100,∞[	[2500, 3000[	[150, 200[	[50, 60[	"sixth"
7	N/A	[3000, 4000[	[200, 500[	[60, 655[	"seventh"
8	N/A	[4000, 16382[	[500, 1500[	[655,∞[	N/A
9	N/A	[16382,∞[	[1500,∞[	N/A	N/A

*NO_DATA: No data received at that time

**N/A: No corresponding valued

OpenXC enables the collection of a set of driving signals, we restrict ourselves to those listed in Table VI. Table VI shows for each variable the range of its values and the frequency of measurement (taken from OpenXc website [40]), as well as the abstract values from the interval domain that we have defined; we do not define any abstraction for the variable *transmition_gear_position* because it is discrete. Now that abstractions have been defined, labels for the graph vertices have to be designed. Table VII shows the labels we have used in this study. For all variable, each abstract value (interval) is mapped to a number from the set $\{-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ as seen in Table VII; and we construct a vertex label as a combination of the variables labels (in the same order presented in Table VII). For instance, a vertex labeled 02000 means that the value of *engine_speed* is in *[500, 1000[* and the *transmission_gear_position* is in "*neutral*" position. The algorithm of Fig.5 is then applied for graphs construction. Considering the anonymity of the vehicle trace files, one driving behavior graph is constructed for data in one vehicle trace file. Hence, we can use the analysis

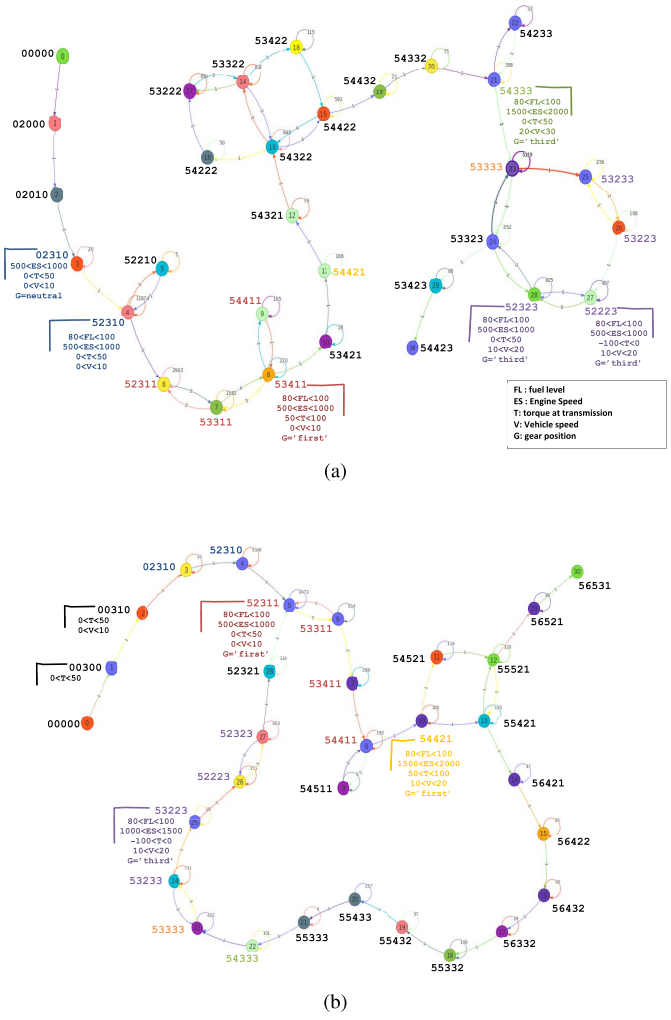

Fig. 11. Example of driving behavior graphs visualization. (a) Driving behavior graph for the Downtown-Crosstown scenario. (b) Driving behavior graph for the East Downtown scenario.

introduced in this section to analyze the similarity between the trace files. We present in Fig.11 two driving behavior graphs created from the traces *"Downtown,Crosstown"* and *"East Downtown"* respectively. For simplicity, we constraint ourselves to graphs of 30 nodes. By visually comparing the two graphs, we can analyze the similarities in the driving behavior of the two drivers. Examples of the information we can get from the comparison are:

- There are some common states between the graphs of the two drivers: The nodes labeled 02310, 52310, 52311, 53311, 53411, 54411, 54421, 54333, 53333, 53233, 53223, 52223, 52323. This means that the drivers have some similarities in the visited states
- The behavior of the driver in Fig.11a is characterized by an increasing in the value of the *engine_speed* whereas an increasing in the value of *torque_at_transmission* is observed in the behavior of the driver in Fig.11b, until they both reach the node 02310.
- As regards the states in common, differences in the weights of the self loop edges is observed. This explained by the fact that the driver spends more time in the

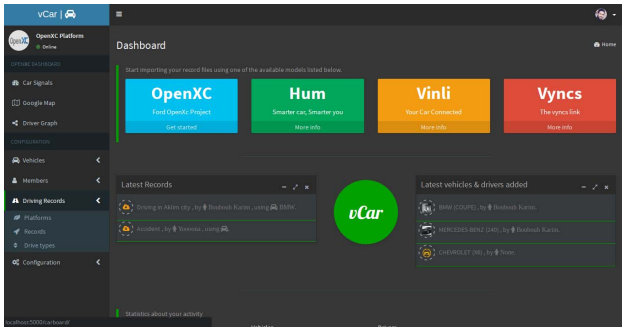

Fig. 12. Home page of the prototype developed "vCar" (*www.vcar.ai*).

nodes with high weighted loop. For instance, the driver in Fig.11b spends more time in the nodes 52311 53411 5411 54421; while the one in Fig.11a tends to spend more time in the nodes labeled 52310, 53311, 54333, 53333.

- Both driver seem to have similar behavior in the node labeled 02310, which is always followed by node 52310; and the node 52311 followed by node 53311.
- While the driver in Fig.11b tends to leave node 52310 to only reach node 52311, the driver in Fig.11a sometimes passes by node 52210. This means that when in node 52310, the value of *torque_at_transmission* for the vehicle of the latter driver may reach a negative value before returning to its positive value and perform shifting.

In a similar way, a more profound analysis can be conducted by analyzing the labels of the two graphs vertices.

It is worth mentioning that the present analysis was implemented in our prototype *vCar* shown in Fig.12. More information about the vCar platform can be found on the website *www.vcar.ai*. *vCar* is a platform for the analysis and visualization of driving data. It is able, for now, to process and visualize data from any trace file in JSON format.

B. Drivers Comparison Based on Graph Matching

Graph matching is a field of graph theory that studies similarity between graphs. This has been widely used in many applications especially for classification and pattern recognition. In this section, we propose the application of graph matching algorithms to mathematically compare the driving behavior graphs, enabling thus a model-based identification of drivers.

The problem of graph comparison [41] can be formulated as finding a mapping $s : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{R}$, given two G and G' from the space of graphs $\mathcal{G}$, such that $s(G, G')$ measures the similarity (or dissimilarity) of G and G'. Different approaches addressing the problem of graph similarity can be found in the literature, we propose the use of the Graph Edit Distance method (GED) for a first study. GED is known for its flexibility, in comparison with other methods as it can be applied to a wide class of graphs [35]. The idea of GED to measure the dissimilarity between two graphs G and G' is to find a suitable sequence of *graph edit operations* (generally, *insertions*, *deletions* and *substitutions* of vertices and edges) that transforms G into G'. This sequence of operations is referred to as an *edit path*. As many edit paths could exist,

TABLE VIII
RESULTS FOR DISSIMILARITY MEASUREMENT OF DRIVING BEHAVIOR GRAPHS

Data Graph (G)			Model Graph (G')			Graph Edit Distance (G,G')
File Name	Nodes	Edges	File Name	Nodes	Edges	
West Downtown 2	30	63	West Downtown	30	60	62.274
	100	275		100	288	293.236
	139	403		102	303	363.419
West Uptown 2	30	61	West Uptown	30	59	76.812
	100	239		100	248	240.047
	176	525		180	571	585.537
Uptown Crosstown 2	30	66	Uptown Crosstown	30	61	70.973
	100	258		100	289	334.525
	129	381		125	417	529.324
East Downtown	30	62	Downtown, Crosstown	30	70	78.369
	100	271		100	254	281.909
	134	430		348	1084	734.423

Model Graph: the model used for comparison
Data Graph: the graph to be compared

a cost $c(e_i)$ associated to each graph edit operation e_i is used. Intuitively, a low cost edit path means that the two graphs are similar. The graph edit distance is therefore defined as:

$$GED(G, G') = \min_{(e_1,e_2,...,e_k)\in P(G,G')} \sum_{i=1}^{k} c(e_i)$$

There exist several algorithms for computing the GED between graphs. Generally, the exact algorithms consider the set of all possible paths as a tree and transform the problem into a tree search algorithm to find the shortest path (path with minimum cost). Authors of [42] have provided a software for graph edit distance computation where five algorithms are implemented (*A**, *Beam Search*, *Munkres Algorithm*, *Hungarian Algorithm* and the algorithm of *Volgenant and Jonker*). The software also provides four similarity kernels from the transformation of the graph edit distance, which makes kernelized learning algorithms applicable to graphs. The software was interfaced with the platform *'vCar'* and was used as a tool to compute the dissimilarity between the driving behavior graphs constructed from vehicle trace files in Table V. The GED algorithm applied in the study is the Hungarian algorithm.

Because of the diversity of driving scenarios and for a more realistic comparison, the algorithm is applied on graphs from similar driving scenarios. The results of graphs matching is presented in table VIII: *"West Downtown 2"* (resp. *"West Uptown 2"*, *"Uptown Crosstown 2"*) file is compared with *"West Downtown"* (*"West Uptown"*, *"Uptown Crosstown"*) file, and *"Downtown, Crosstown"* file is compared with *"East Downtown"*. To evaluate the relationship between the GED distance and the size of the graph, GED distance was computed for smaller parts of the files (graphs of 30 nodes, 100 nodes)), and then for all the file. Results show that the value of GED increases when the number of nodes and edges is larger. This means that the similarity between driving behaviors decrease when more data is recorded, which is quite predictable as more nodes and edges are created and linked when more data is available. Computing GED distances between driving behavior graphs is also important for the graphs clustering which will be very useful for drivers styles classification as well as drivers identification.

VI. Conclusion

Driving behavior analytics solutions are among the solutions that are gaining more attention recently. Using a set of powerful statistical tools and machine learning techniques, they are able to analyze driving data and provide advanced reports on driving behavior. Their applications are of great benefit for individual drivers to keep track of their driving performance as well as for companies and decision makers to evaluate and compare the driving behavior of multiple drivers. In this paper, we presented two main analyses of driving behavior that of (1) an automatic verification of the driving behavior safety and (2) comparative analysis of driving behavior. As a first step of both analysis, the numerical abstraction using the interval domain was used to prepare data for analysis. Then for modeling, Probabilistic Rectangular Hybrid Input/Output Automata (PRHIOA) and Labeled Directed Graphs were used to create a data-based model of the driving behavior. The safety analysis of driving behavior was carried out by applying the model checking of the driving behavior represented as PRHIOA, while the comparative analysis were conducted by using graph matching techniques applied on the graph representation of the driving behavior. In addition to their mathematical basis, the models were useful for a better driving behavior visualization.

One major conclusion of this paper is that the abstraction represents a key step as it influences not only the size of the model but also the accuracy of the analysis. In model checking for example, although some of the properties presented in this paper may not be accurately parametrized, the results of their implementation show how much a bad abstraction can fail to notice critical information about the driver behavior. Therefore, besides costs, the definition of a convenient abstraction must be with consideration of the parameters of the properties that need to be expressed. It is also important to mention that the present paper uses a static abstraction where the intervals of the abstract domain are set in advance. However, a dynamic abstraction can be considered where machine learning algorithms are used to retrieve the suitable intervals from data.

Acknowledgments

The authors would like to thank Professor Kazuya Takeda, Nagoya University, for the interesting discussions that we have shared, and also for providing us with the driving data used in this research. Professor Takeda is a member of Acoustic Society of Japan (ASJ), the Institute of Electronics, Information, and Communication Engineers (IECE) Japan, Information Processing Society Japan (IPSJ) and IEEE. He is also an executive board member of ASJ and an associate editor of IEICE Transaction on System and Information. They are also immensely grateful to the anonymous reviewers for their constructive comments.

References

[1] K. Madia. *Big Data on Wheels*. Accessed: Apr. 20, 2016. [Online]. Available: http://www.ibmbigdatahub.com/blog/big-data-wheels

[2] E. Simoudis. *The Automotive Industry's Big Data Challenge*. Accessed: Apr. 24, 2016. [Online]. Available: https://www.enterpriseirregulars.com/104636/

[3] i4drive. *Advanced Driving Experience*. Accessed: Nov. 16, 2016. [Online]. Available: https://www.i4drive.com/driving-analytics

[4] CLOUD MADE. *Cloud Made: Car & Driver Analytics*. Accessed: Nov. 25, 2016. [Online]. Available: http://cloudmade.com/solutions/car-driver-analytics

[5] Zendrive. *Zendrive: Making Roads Safe Using Data and Analytics*. Accessed: Sep. 18, 2016. [Online]. Available: https://www.zendrive.com

[6] AMODO. *Amodo: Advanced Driving Behavior Analytics*. Accessed: Nov. 22, 2016. [Online]. Available: https://www.amodo.eu

[7] A. A. Marzooqi, "Road safety system for monitoring fleet drivers," in *Proc. 68th Road Safety Congr. Held Blackpool, Safer Driving, Reducing Risks, Crashes Casualties*, Mar. 2003, p. 12.

[8] T. A. Dingus *et al.*, "The 100-car naturalistic driving study, phase II—Results of the 100-car field experiment," Virginia Tech Transp. Inst., Blacksburg, VA, USA, Tech. Rep. DOT HS 810 593, Apr. 2006. [Online]. Available: https://www.bsse.ethz.ch/mlcb/education/previous-courses/computational-approaches-10.html

[9] K. Takeda, J. H. Hansen, P. Boyraz, L. Malta, C. Miyajima, and H. Abut, "International large-scale vehicle corpora for research on driver behavior on the road," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1609–1623, Dec. 2011.

[10] A. Bender, G. Agamennoni, J. R. Ward, S. Worrall, and E. M. Nebot, "An unsupervised approach for inferring driver behavior from naturalistic driving data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3325–3336, Dec. 2015.

[11] B. Higgs and M. Abbas, "Segmentation and clustering of car-following behavior: Recognition of driving patterns," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 81–90, Feb. 2015.

[12] G. A. M. Meiring and H. C. Myburgh, "A review of intelligent driving style analysis systems and related artificial intelligence algorithms," *Sensors*, vol. 15, no. 12, pp. 30653–30682, 2015.

[13] K. Kircher, A. Larsson, and J. A. Hultgren, "Tactical driving behavior with different levels of automation," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 158–167, Feb. 2014.

[14] A. M. Noble, T. A. Dingus, and Z. R. Doerzaph, "Influence of in-vehicle adaptive stop display on driving behavior and safety," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2767–2776, Oct. 2016.

[15] N. Oliver and A. P. Pentland, "Driver behavior recognition and prediction in a smartcar," in *Proc. Int. Soc. Opt. Photon. (AeroSense)*, 2000, pp. 280–290.

[16] A. Sathyanarayana, P. Boyraz, and J. H. Hansen, "Driver behavior analysis and route recognition by hidden Markov models," in *Proc. IEEE Int. Conf. Veh. Electron. Safety (ICVES)*, Sep. 2008, pp. 276–281.

[17] N. Dapzol, "Driver's behavior modeling using the hidden Markov model formalism," in *Proc. Young Researcher Seminar Eur. Conf. Transp. Res. Inst.*, 2005, pp. 1–14.

[18] M. Van Ly, S. Martin, and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2013, pp. 1040–1045.

[19] Y. Liao, S. E. Li, G. Li, W. Wang, B. Cheng, and F. Chen, "Detection of driver cognitive distraction: An SVM based real-time algorithm and its comparison study in typical driving scenarios," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 394–399.

[20] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, "Behavior classification algorithms at intersections and validation using naturalistic data," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 601–606.

[21] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *Privacy Enhancing Technol.*, vol. 2016, no. 1, pp. 34–50, 2016.

[22] A. Wahab, T. C. Keong, H. Abut, and K. Takeda, "Driver recognition system using FNN and statistical methods," in *Advances for in-Vehicle and Mobile Systems*. Boston, MA, USA: Springer, 2007, pp. 11–23.

[23] C. Miyajima *et al.*, "Driver modeling based on driving behavior and its evaluation in driver identification," *Proc. IEEE*, vol. 95, no. 2, pp. 427–437, Feb. 2007.

[24] S.-W. Chen, C.-Y. Fang, and C.-T. Tien, "Driving behaviour modelling system based on graph construction," *Transp. Res. C, Emerg. Technol.*, vol. 26, pp. 314–330, Jan. 2013.

[25] P. Cousot and R. Cousot, "Basic concepts of abstract interpretation," in *Building the Information Society*. Boston, MA, USA: Springer, 2004, pp. 359–366.

[26] P. Cousot and R. Cousot, "Abstract interpretation: Past, present and future," in *Proc. Joint Meeting 23rd EACSL Annu. Conf. on Comput. Sci. Log. (CSL), 29th Annu. ACM/IEEE Symp. Log. Comput. Sci. (LICS)*, Jul. 2014, p. 2.

[27] Wikipédia. (2016). *Vitesse Maximale Autorisée Sur Route en France—Wikipédia, L'encyclopédie Libre*. [Online]. Available: http://fr.wikipedia.org/w/index.php?title=Vitesse_maximale_autoris%C3%A9e_sur_route_en_France&oldid=131825962
[28] K. Wagstaff *et al.*, "Constrained k-means clustering with background knowledge," in *Proc. ICML*, vol. 1, 2001, pp. 577–584.
[29] E. Romera, L. M. Bergasa, and R. Arroyo, "Need data for driver behaviour analysis? Presenting the public UAH-driveset," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 387–392.
[30] A. Bouhoute, R. Oucheikh, Y. Zahraoui, and I. Berrada, "A holistic approach for modeling and verification of human driver behavior," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2015, pp. 1–7.
[31] A. Puri and P. Varaiya, "Decidable hybrid systems," *Math. Comput. Model.*, vol. 23, no. 11, pp. 191–202, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0895717796000726
[32] K. S. Narendra and M. A. Thathachar, *Learning Automata: An Introduction*. New York, NY, USA: Dover, 2012.
[33] R. P. Singh, "Application of graph theory in computer science and engineering," *Int. J. Comput. Appl.*, vol. 104, no. 1, pp. 10–13, 2014.
[34] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Anal. Appl.*, vol. 13, no. 1, pp. 113–129, Feb. 2010.
[35] L. Livi and A. Rizzi, "The graph matching problem," *Pattern Anal. Appl.*, vol. 16, no. 3, pp. 253–283, Aug. 2013.
[36] C. Olaverri, "Behavior signal processing laboratory [ITS research lab]," *IEEE Intell. Transp. Syst. Mag.*, vol. 8, no. 1, pp. 72–76, 2016.
[37] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. Int. Conf. Comput. Aided Verification*. Berlin, Germany: Springer, 2011.
[38] K. Igarashi, K. Takeda, F. Itakura, and H. Abut, "Is our driving behavior unique?" in *DSP for in-Vehicle and Mobile Systems*. Boston, MA, USA: Springer, 2005, pp. 257–274.
[39] OpenXC. *Vehicle Trace Files*. Accessed: May 30, 2016. [Online]. Available: http://openxcplatform.com/resources/traces.html
[40] OpenXC. *Vehicle Trace Files*. Accessed: May 30, 2016. [Online]. Available: http://openxcplatform.com/about/data-set.html
[41] K. Borgwardt and O. Stegle, "Computational approaches for analyzing complex biological systems," Univ. Tübingen, Tübingen, Germany, 2010.
[42] K. Riesen, S. Emmenegger, and H. Bunke, "A novel software toolkit for graph edit distance computation," in *Graph-Based Representations in Pattern Recognition*. Berlin, Germany: Springer, 2013, pp. 142–151.

Afaf Bouhoute received the bachelor's degree in computer science and the master's degree in information system, networking and multimedia from the Faculty of Science, Sidi Mohamed Ben Abdellah University, Fes, Morocco, in 2010 and 2012, respectively, where she is currently pursuing the Ph.D. degree. She is also a Teaching Assistant with the National School of Applied Sciences of Fes, Morocco. Her research interests include vehicular ad-hoc networks, driver behavior modeling, and analytics solutions for intelligent transportation.

Rachid Oucheikh was born in Skoura, Morocco, in 1989. He received the baccalaureate degree in electronics and the State Engineering Diplomate degree in computer sciences and networks from the National Institute of Posts and Telecommunications, Rabat, Morocco, in 2007 and 2012, respectively. He is currently pursuing the Ph.D. degree with the Faculty of Science, Sidi Mohamed Ben Abdellah University, Morocco. His current research interests include real-time systems verification, constraints satisfaction problems, and driver behavior modeling.

Karim Boubouh is currently pursuing the Ph.D. degree with the Faculty of Science, Sidi Mohamed Ben Abdellah University, Fes, Morocco, under the supervision of Prof. I. Berrada. His research is focused on driver behavior identification using deep learning algorithms.

Ismail Berrada received the Ph.D. degree in computer science from the University Bordeaux 1, France, in 2005. He was an Assistant Professor with the University of La Rochelle for 5 years. He is currently an Associate Professor of computer science with the Faculty of Science, Sidi Mohamed Ben Abdellah University, Fes. His research interests include cognitive radio networks, radio resource management, vehicular ad hoc network, and software testing and verification.