

Analyzing Driver Behavior Compliance under HoS Regulations*

Ignacio Vellido-Expósito, Juan Fernández-Olivares^a, Raúl Pérez^b and Luis Castillo^c

Department of Computer Science and AI, University of Granada, Andalusian Research Institute in Data Science and Computational Intelligence, Granada, Spain

Keywords: Fleet Telematics, Regulation Compliance, Artificial Intelligence, Automated Planning.

Abstract: World wide spreaded Hours of Service (HoS) regulations constraint drivers' amount of working and driving time without resting. Transport companies are extremely interested on interpreting what their drivers are doing, based on the raw information of event logs generated by fleets' onboard devices, considering the terms defined by HoS regulations. This work addresses the problem of analyzing the compliance of a driver wrt HoS, by using AI techniques to classify and label the information of a driver's log according to HoS terms. The final result is a human-interpretable descriptive model of driver behaviour that leverages the company situational awareness, empowering staff responsible of operations to make better informed decisions.

1 INTRODUCTION

A problem of paramount importance in transport companies is to determine whether driver activity conforms with Hours of Service (HoS) regulation to forestall illegal behavior and avoid costs due to sanctions. HoS regulation are imposed by world wide transport authorities (Meyer, 2011; Goel and Vidal, 2013), which constraint the amount of drivers' working, driving and resting time when delivering a transport service. As a consequence, scheduled driving plans have to be aligned with laws that define the legal behavior of a driver.

The widespread adoption of onboard IoT devices in vehicle fleets enables recording the activity of drivers in event logs. Moreover, transport companies are encouraged (currently almost forced by law) to endow every transport asset with electronic sensors (i.e. tachographs) which record the different activities a driver performs when delivering a transport service. This context enables driver activity to be recorded in event logs and enhances the need for driving work plans to be monitored and analyzed either by companies' decision makers or authorities to analyze drivers' legal compliance. Since raw event logs gen-

erated by onboard fleet devices are difficult to be directly interpreted, an important technical challenge is to come up with easily interpretable descriptive models that help understand the huge amount of information stored in such event logs.

In response to this need, the paper presents a method that starting from a raw, real event log extracted from a tachograph, 1) classifies and tags activities, and 2) generates a human-interpretable descriptive model that consists of an extension of the original raw log with additional attributes that, after the classification and labeling process, characterize each and every activity in the log.

The characterization provides a semantics to the raw log by extending it with additional information about aspects such as the type of activity according to HoS regulation, the type of HoS sequence/subsequence the activity belongs to, as well as whether the activity (and hence the sequence which it is involved in) is compliant with the HoS regulation. The method is validated with an experimentation over real data and by a proof of concept.

This method provides several advantages. On the one hand, operational staff can determine at a glance information about the legality of driver's behavior, without the need to resort on a burdensome inspection of the raw log. On the other hand, the extended log (descriptive model) incorporates information with sufficient granularity to interpret what each of the drivers is doing considering HoS regulations. In summary, the labeled event log can be easily interpreted

^a  <https://orcid.org/0000-0002-7391-882X>

^b  <https://orcid.org/0000-0002-1355-1122>

^c  <https://orcid.org/0000-0002-4910-7752>

*This work is being partially funded by the Andalusian Regional Project B-TIC-668-UGR20 and the Spanish National Project RTI2018-098460-B-I00 with FEDER funds.

HoS CONCEPT	TYPE	DURATION		ADDITIONAL CONSTRAINTS
		MIN	MAX	
NO BREAK		0	<15min	Always part of a basic sequence
BREAK	BREAK_T2	15min	<30min	
BREAK	BREAK_T3	30min	<45min	
BREAK	BREAK_T1	45min	<3hrs	
DAILY REST	REDUCED	9hrs	<11hrs	Only 3 times a week, Split into 2 parts of at least 3 an 9 hours of duration.
DAILY REST	NORMAL	11hrs	24hrs	
WEEKLY REST	REDUCED	24hrs	45hrs	Only once in two consecutive weeks. Compensate with same duration before the end of the third week following the week considered.
WEEKLY REST	NORMAL	45hrs		

HoS CONCEPT	TYPE	ACCUMULATED DURATION		ADDITIONAL CONSTRAINTS
		MIN	MAX	
BASIC SEQUENCE		Composed by any combination of [Driving Other Pause Idle] whenever <i>Pause</i> is NO BREAK (<i>duration</i> < 15 min)		
DRIVING PERIOD	CONTINUOUS	0	<=4.5hrs	Basic Sequence ended by a break_t1 or a Daily Rest.
DRIVING PERIOD	SPLITTED	0	<=4.5hrs	Driving period splitted by two breaks : one break_t2 and then another break_t3
DAILY DRIVING	NORMAL	0	<=9hrs	
DAILY DRIVING	EXTENDED	>9hrs	<=10hrs	Only 2 times a week.
WEEKLY DRIVING		0	<=56hrs	

Figure 1: A summary of HoS concepts, duration and additional constraints. We make use of "<" and "<=" symbols to indicate whether the allowed duration is represented by either an open or closed interval, respectively.

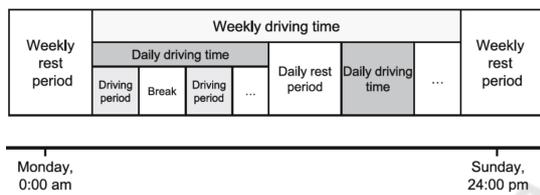


Figure 2: Hierarchical relations and structure of the different types of sub-sequences, extracted from (Meyer, 2011).

by company experts which can make more informed decisions considering either the historical or current labeled situation of a driver.

The remainder of this paper shows, firstly, a detailed description of the novel problem addressed and the main contribution of the approach (Section 2). Then, we outline the overall methodology (Section 3), and explain some background concepts needed to ease the overall description of the approach (Section 4). The technical details are shown in Sections 5 and 6. We end with an experimentation conducted over a proof of concept of the application (Section 7), and discuss related and future work (Sections 8 and 9).

2 HoS REGULATION

Transport companies are interested on making decisions to govern the behavior of their drivers by predicting whether a driver is close to committing a regulation violation, but previous to this they have to face the challenge of characterizing drivers according to their driving style with respect to the HoS regulation. With the incorporation of IoT onboard devices, companies are forced to deal with large data volumes of daily event logs, with thousands of events, that need to be interpreted with respect to HoS regulations. Therefore, there is a clear need for an expert to directly interpret what a driver has been doing during the period of time encompassed by a given event log.

Concretely, every event in an activity log is a tuple $(a, start, end, dur, d_id)$, where each component refers to: activity identifier (a), event start ($start$) and end (end) times, event duration (dur), and driver identifier (d_id), respectively. A value for a is any of the labels [Driving, Other, Pause, Idle] meaning that the driver is either Driving, performing Another Work, at Pause or Idle during dur minutes, between $start$ and end . Our goal is to provide semantics to each event, by completing this raw information with additional attributes whose values are the terms provided by the HoS regulation, which are detailed in the following.

This regulation has been extensively analyzed in (Goel and Vidal, 2013; Meyer, 2011). It is applied in several countries, but in this work we focus on the European Union regulation (EC) No 561/2006 (Meyer, 2011). The basic terms refer to four types of driver activities as *break* (short period for recuperation), *rest* (period at driver free disposal with enough time to sleep), *driving* (time during which the driver is operating a vehicle) and *other work* (time devoted to any work except driving, like loading). These activities do not exactly correspond to the event labels above defined, but they are defined according to such events and their attributes. That is to say, the regulation defines different types of *rest and break periods* (see Figure 1, left-hand table) according to the duration of a *Pause* event. Moreover, there are different types of *driving sequences*. A *basic driving sequence* is composed of a totally ordered set of the elements of [Driving, Pause, Other, Idle] constrained to the duration of any *Pause* is less than 15 minutes (see Figure 1, right-hand table for more types). More constraints are defined over the duration of the rests and breaks, and over the accumulated duration of driving sequences (see both tables in Figure 1). The regulation provides a set of basic and optional rules, should the former not be satisfied, thus allowing more flexibility to generate and interpret driving schedules under such constraints. For example, either a break of 45 mins has to

be taken after 4.5 hours of accumulated driving or the break can be taken splitted in two parts of at least 15 mins and 30 mins respectively. This feature is good for drivers since it provides flexibility to their work, but complicates the interpretability of what they are doing. Regulations also define additional constraints (for example, the maximum number of occurrences of a reduced rest in a weekly driving period), or the hierarchical relationship between the different types of sub-sequences, as well as their internal structure (see Figure 2).

2.1 The Problem

In summary, *the problem consists of classifying and labeling events of a given log by considering the legal terms above described about HoS regulations*. Concretely, the observed behavior has to be tagged in terms of driving periods, daily driving periods (and their types), and weekly driving periods (and their types) as shown in Figures 1 (right-hand table) and 2. Last but not least, identified subsequences have to be tagged as legal or illegal according to their compliance with HoS regulation.

The main contribution of this work consists of addressing the above described labeling of activities and classification of illegal sequences as a parsing process based on automated planning (AP) techniques, as well as an experimentation that validates our approach. AP is a widely known field of Artificial Intelligence devoted to develop algorithms, called planners, that implement search-based problem solving processes to determine a plan (sequence of actions) for a situated agent to achieve a goal. The inputs provided to a planner are a model of actions that describes the behavior of the agent, called a planning domain, and, a planning problem, a description of the current state of the agent and the goal to be achieved.

The main use of these techniques is the generation of plans, but recent works have shown their usefulness to recognize sequences of actions. This last use lays on the fact that when a planner is provided with an initial state that incorporates a sequence of observations representing the past activities of an agent, and a repertory of possible goals the agent is intended to achieve, the planning process can be used to determine which of the possible goals is intended to be achieved by the agent executing the observed actions. This view of automated planning is called plan recognition and it is the one followed by our approach.

Concretely we are using in this work temporal hierarchical task network planning techniques (temporal HTN), a variant of AP where the planning domain is represented as a hierarchical knowledge base

in which the behavior of the agent is defined in terms of a compositional hierarchy of tasks/subtasks where primitive tasks represent temporally annotated actions to be executed by the agent, and compound tasks represent temporal ordering strategies of actions. The key aspect in HTN is that compound task can be decomposed in alternative ways by means of methods. A method is a pair (t, d) that describes one way to achieve the task t is to perform the tasks specified in the *task network* d , a temporally constrained set of tasks/subtasks representing things that need to be done. The hierarchical planning process is based on a search process that starts at a given top-level task network, and recursively reduces it by opportunistically applying methods. The current task network is transformed step by step in each reduction by inserting tasks according to the order and temporal constraints specified in the methods, until it consists of a plan only composed of primitive actions.

The reason to use a temporal and hierarchical planning approach is two-fold: on the one hand, the event log observations are temporally annotated and constrained, hence we need to resort on a technique able to manage such temporal constraints. On the other hand, the HoS regulation can be naturally represented as a hierarchy of temporal tasks.

In our case a given event log represents a sequence of temporally annotated observations and the planning process 1) identifies different temporal sub-sequences of a driver's daily and weekly driving activity, 2) labels them according to the terms defined by HoS regulation, and 3) classifies them as legal/illegal. It is important to note that the identification of temporal sub-sequences is a process analogous to grammar parsing, and in our case is addressed as a knowledge-driven search process with two main features: i) it is implemented in a hierarchical planner, ii) the planner is provided with a knowledge-based model representing the HoS regulation. The parsing task is performed over the entire event log considering that the decomposition methods provided in the HTN model can be seen as a set of production rules of an attribute grammar.

In the following sections we describe the general steps of our method, provide a description of the knowledge-based representation of HoS regulation, then we briefly describe the knowledge-driven search process to parse the log, and the experimentation and proof of concept carried out to validate our approach.



Figure 3: A schema of the methodology of the approach.

3 OVERALL METHODOLOGY

The methodology we have followed to provide a solution to the problem consists of the following steps (see Figure 3):

1. Formalizing HoS rules as productions of an attribute grammar.
2. Transforming the raw event log into a sequence of temporal observations.
3. Parsing the sequence of temporal observations by using the attribute grammar representing the HoS regulation.
4. Generating a human-interpretable descriptive model that consists of an extension of the original event log with additional attributes

The following sections describe in detail these steps.

4 HoS AS AN ATTRIBUTE GRAMMAR

We have used attribute grammars (Knuth, 1968) as an intermediate representation to formalize the description of the HoS regulation in Europe. The main reason is that attribute grammars, apart from being an excellent mechanism to describe context-sensitive grammars, allow to easily represent the required temporal and numeric constraints with grammar attributes. An attribute grammar extends productions with *semantic rules*¹ of the form $\{\langle condition \rangle \langle assignments \rangle\}$, where $\langle condition \rangle$ defines the applicability conditions of the production and $\langle assignments \rangle$ define how synthesized attributes of parent nonterminals (at left-hand side of the production) are calculated from attributes of the right-hand side of the production. In the following we describe the production rules to recognize basic breaks, rests and basic sequences (as described previously).

¹We are using an *ad hoc* syntax for attribute grammar in order to ease the explanation.

4.1 Recognizing Breaks, Rests and Basic Sequences

The terminals of the grammar are the labels $[D, O, P, I]$, since they are used to distinguish the types of observations which may be found in an event log (*Driving, Other, Pause, Idle*). Terminals have associated attributes such as *duration* that are represented following the "dot" notation. For example $P.dur$ represents the duration of a concrete, recognized *Pause* event in the event log. On the other hand, every non-terminal has associated two attributes that represent the *driving time* (attribute dt) and *pausing time* (attribute rt) of the current recognized token. For example, $b.t1.rt$ represents the pausing time of the token $b.t1$ which in turn represents a *Break of Type 1* (see BREAK_T1 in Figure 1).

Breaks Recognition. The pausing time of any pause event in the log is further used to discriminate between different types of Breaks or Rests, according to rules in Figure 1 (left-hand table). For example, a production that recognizes a Break of Type 1, is represented as follows

```
b_t1 : P {(and (P.dur >= 45min) (P.dur < 3hrs)
          (b_t1.rt := P.dur)}
```

describing that a Break of Type 1 corresponds to a pause greater than 45 minutes and lesser than 3 hours, and that the duration of such pause event $P.dur$ is assigned as the duration of the break $b.t1.rt$. Other types of breaks are represented in a similar way with grammar productions.

Rests Recognition. Daily and weekly rests, represented by the non-terminal symbols rd and rw , may be normal or reduced, and are defined according to the duration of pauses described in Figure 1 (left-hand table). For example, the following productions represent this aspect of the regulation

```
rd : rd_normal {rd.rt := rd_normal.rt};
rd : rd_reduced {rd.rt := rd_reduced.rt};

rd_normal : P {(and (P.dur >= 11h) (P.dur < 24h)
                  (rd_normal.rt := P.dur) );
```

The first production describes the legal types of daily rests, while the second represents that a normal daily rest is a pause with a duration between 11 and 24 hours.

Basic Sequences Recognition. Figure 1 (right-hand table) shows that a basic sequence is composed of any number of activities such that the duration of any *Pause* is strictly less than 15 mins. Firstly, we need to define the legal activities allowed, which is represented by the non-terminal symbol `elt` and the following productions:

```
elt: D {(elt.dt := D.dur;elt.rt:= 0)};
elt: P {(P.dur < 15mins);
      (elt.dt := 0; elt.rt:= P.dur)};
elt: O {(elt.dt := elt.dt := 0)};
elt: I {(elt.dt := elt.dt := 0)};
```

Then we can make use of the following recursive production to define what a correct basic sequence is. Moreover, it defines that the driving time of a sequence is the accumulated driving time of its components. Note that in attribute grammars occurrences of the same symbol are differentiated by subscripts to be correctly identified.

```
baseq : elt {(baseq.dt:= elt.dt)};
baseq : elt baseq2 {(baseq.dt := elt.dt+baseq2.dt)};
```

Finally the production rules to recognize the remaining HoS concepts described in Figure 1 are represented following the same principles illustrated previously.

5 GENERATING TEMPORAL OBSERVATIONS

Every record of the event log is translated into an observation ($a, i, type, start, end, dur, d_id$) where i is an index representing the order in the sequence of events, a is a unique identifier for the action, $type$ indicates the type of action (one of $[D, O, P, I]$ corresponding to the actions labels $[Driving, Other, Pause, Idle]$), d_id is the identifier of the driver, and $start, end, dur$ represent the start and end times of the action and its duration, respectively. For example the event representing a *Pause* of 68 minutes from 15:46 to 16:54 on 05/01/2017 would be represented as follows:

```
Event :
(Pause "05/01/2017 15:46"
 "05/01/2017 16:54" 68 driver1)
Observation:
(P15 39 P "05/01/2017 15:46"
 "05/01/2017 16:54" 68 driver1)
```

6 PARSING PROCESS

Algorithm 1: Algorithm for classifying and labeling based on event log parsing.

Input: \mathcal{G} (productions), \mathcal{S} (observations)

```

1  $\mathcal{A} \leftarrow [top\_level\_symbol]$  /*the agenda*/;
2  $\Pi \leftarrow []$  /*empty plan*/;
3  $i \leftarrow 1$  /*index of current_event*/;
4  $current\_event \leftarrow \mathcal{S}[i]$ ;
5 Initialize contexts (Week, Day, DayType,
   SeqOrder, SeqType, Token, Legal);
6 while  $\mathcal{A}$  not empty do
7   extract the first element  $t$  from  $\mathcal{A}$ ;
8   if  $t$  is TERMINAL then
9     if  $type\_of\_action(current\_event)$  is in  $[D O$ 
10       $P I]$  then
11       if not  $Check\_temporal\_constraints()$ 
12        then
13         Insert event in  $\Pi$  as ILEGAL
14         temporal action
15       else
16         Insert event in  $\Pi$  as temporal
17         action;
18         Label the action with contexts;
19          $current\_event \leftarrow \mathcal{S}[i++]$ ;
20     else
21       Insert in  $\Pi$  event as ILEGAL
22   if  $t$  is NONTERMINAL then
23      $LEGAL \leftarrow YES$ ;
24     Update contexts considering  $t$ ;
25     if  $t$  is  $UPDATE_t$  then
26       update  $t.att$  according to  $t.successor$ 
27     else
28       Choose from  $\mathcal{G}$  one production  $t:rh_i$ 
29       not yet processed ( $rh_i = \{t_1 .. t_n\}$ );
30       if All productions already processed
31       then
32          $LEGAL \leftarrow NO$ 
33       else
34          $t.successor \leftarrow rh_i$ ;
35          $\mathcal{A} \leftarrow \mathcal{A} \cup \{t_1 .. t_n UPDATE_t\}$ 
36   return  $\Pi$  with the labeled and classified actions
```

The algorithm of the parsing process is shown in Algorithm 1. The algorithm receives as input the set of productions representing the HoS regulation (the grammar \mathcal{G}) and the set of temporal observations \mathcal{S} (generated as above explained), and returns a plan Π that contains the temporal observations transformed in actions which are labeled with appropriate values for the set of extended attributes $[week\ day\ DayType\ SeqOrder\ SeqType\ Token\ Legal]$ whose semantics is described in subsections below. The first steps of the algorithm consist of initializing the agenda \mathcal{A} , that is a list containing the pending productions (or compound tasks) to be processed, the plan Π that is incremen-

tally generated as events are processed, the index i that refers to the current event to be processed in the log, and the extended attributes.

The main loop of the algorithm explores the set of productions of the grammar G trying to match grammar productions with the current event of the log (indexed by the variable i). If for the current event to be processed a matching with a terminal of a legal production is found, the event is transformed in an action, added to the plan Π (if the temporal constraints of the temporal observation are consistent with the current plan), and labeled accordingly. This parsing process is carried out over the entire set of observations in \mathcal{S} . The main steps are described in the following sections.

6.1 Event Recognition

Each *type* of event [*Driving, Other, Pause, Idle*] has an associated primitive temporal action $\langle type \rangle_p$ in the domain, that is instantiated according to event information and added to the plan Π when the parsing has processed an event of that type. Therefore, recognizing a concrete event that is at any given position i with label *type* consists of adding its corresponding primitive action to Π such that (i) the temporal points of the action are consistent with the temporal information of the event, and (ii) guaranteeing that the temporal constraints of a are consistent with the rest of temporal constraints of the actions already added to the plan. These conditions are checked by the planning process itself.

The primitive tasks represent the "token" (for example, a *Pause*) that has been read, and the algorithm describes in essence in Line 13 that a primitive task (for example of type *Pause*) has to be added to the plan Π when there is an observation of the same type at the reading pointer position. Interestingly, the ability to represent temporal constraints on tasks at any level of the hierarchy makes possible to represent that the start and end points of the primitive action are constrained by the temporal information of the observation (described in Section 5). This is the point where temporal planning capabilities play a central role since the algorithm will fail to classify the action as ILEGAL (Line 11) if the temporal constraints cannot be met. If the primitive task is successfully inserted, the action is classified as LEGAL, and the index of current event to be processed moves to the next position. The labeling of actions performed in Line 14 is explained in the following.

6.2 Labeling of Events

The labeling process accounts for the different time resolution contexts, defined by HoS regulation, which an event may belong to. Because of that, every event in the event log is annotated with six labels² corresponding with six additional attributes with provide semantics to every event according to the following contexts: *Week* (integer values), *Day* (integer values) *DayType* (with possible values *ndd* or *edd*, corresponding to the concepts of *normal daily driving* and *extended daily driving* shown Figure 1), *SeqOrder* (representing the order of appearance of a sub-sequence as *first*, *second* or *third*), *SeqType* (representing if an action belongs to a sub-sequence of type *split* or *continuous* see Figure 1), *token* (representing if the action is a break or a driving action with possible values according to the types of breaks in Figure 1 with the possible values *b.t1*, *b.t2*, *b.t3*, etc.). Finally, the last attribute *Legal* describes if the activity has been classified by the event recognition process as legal or illegal.

We have made extensive use of the features of the attributes of every production in order to label records of the event log. On the one hand, in Line 28 the set of symbols for every non-terminal τ is extended with the special symbol $\{UPDATE_\tau\}$ for the algorithm to be able to appropriately update the attributes of non-terminals (Line 20). Moreover, a sequence is identified as ILEGAL (Line 26) when all the productions for its non-terminal symbol have been processed and no UPDATE symbol has been reached. On the other hand, the set of attributes of every non-terminal is extended with as many attributes as described above.

7 EXPERIMENTATION

We have validated our methodology with an experimentation using real tachograph logs provided by an industrial collaborator. We were provided with a dataset formed by two-weeks-long sequences of activities from 290 different drivers.

Our experimentation consisted of splitting the dataset by driver, transforming each associated log into a set of observations. Additionally the algorithm has been implemented using an off-the-shelf HTN planner. The planner is provided with an HTN domain that has been written by translating the productions of the grammar representing the HoS regulation, and setting the goal as recognising each action while outputting the activities with the appropriate contexts.

²We are assuming six labels and a reduced set of possible values for each label to ease the explanation.

Table 1: Labelling output after processing a daily sequence. This example shows an extended driving period where the first and second sequences contains split rests and the third is taken uninterrupted. The value "A" in the Token column represents any activity other than a Pause.

Original Log					Annotated Labels						
Driver	Start	End	Duration	Activity	Week	Day	DayType	Sequence	BreakType	Token	Legal
driver3	12/01/2017 09:55	12/01/2017 09:57	2	Driving	3	6	edd	first	split_1	A	yes
driver3	12/01/2017 09:57	12/01/2017 10:41	44	Break						B.T2	yes
driver3	12/01/2017 10:41	12/01/2017 10:43	2	Driving					split_2	A	yes
driver3	12/01/2017 10:43	12/01/2017 11:57	74	Break						B.T3	yes
driver3	12/01/2017 11:57	12/01/2017 12:00	3	Driving				second	split_1	A	yes
driver3	12/01/2017 12:00	12/01/2017 12:06	6	Break						B.T0	yes
driver3	12/01/2017 12:06	12/01/2017 12:08	2	Driving						A	yes
driver3	12/01/2017 12:08	12/01/2017 12:36	28	Break						B.T2	yes
driver3	12/01/2017 12:36	12/01/2017 13:07	31	Driving					split_2	A	yes
driver3	12/01/2017 13:07	12/01/2017 13:13	6	Break						B.T0	yes
driver3	12/01/2017 13:13	12/01/2017 15:31	138	Driving						A	yes
driver3	12/01/2017 15:31	12/01/2017 15:39	8	Break						B.T0	yes
driver3	12/01/2017 15:39	12/01/2017 15:59	20	Driving				third	uninterrupted	A	yes
driver3	12/01/2017 15:59	12/01/2017 18:00	121	Break						B.T3	yes
driver3	12/01/2017 18:00	12/01/2017 18:54	54	Driving						A	yes
driver3	12/01/2017 18:54	12/01/2017 19:08	14	Break						B.T0	yes
driver3	12/01/2017 19:08	12/01/2017 19:09	1	Driving						A	yes
driver3	12/01/2017 19:09	12/01/2017 19:14	5	Other						A	yes
driver3	12/01/2017 19:14	13/01/2017 05:16	602	Break						DR.T2	yes

Once the system returned the labelled log, we selected multiple sequences at random, both legal and illegal, and manually verified that the output was the appropriate under the HoS regulation.

The additional labels provide an exact summarisation of the driver behaviour according to the HoS regulation, resulting in a human interpretable descriptive model, much more readable by experts and consequently enabling business decisions based of tachograph data, like assigning drivers to routes according to their driving styles or identifying problematic tendencies. Furthermore, because the system is capable of identifying illegal sequences, a driver could obtain immediate feedback after committing an infraction, and therefore forestall future action that could result in sanctions for the company.

Table 1 shows an output example for a daily driving sequence. As we can see, the original data is not easily interpretable, since they are formed by different patterns of *Driving*, *Other* and *Break* actions of distinct duration. However, thanks to the annotated labels the output becomes more informative and understandable, giving an overview of the driver behaviour in multiple levels of granularity. In our example, it becomes more explicit that the driver carried out an extended daily driving with three sequences, where the break of the first and second are taken in two parts, and the break of the third sequence is taken uninterruptedly at the end. Additionally, the last two columns provide further information of how each action is considered under the regulation, and that none of them is illegal.

Another example is shown in Table 2, displaying the response of our system against non legal sequences. As they do not fit correctly under the HoS regulation, some contexts cannot be recognised and they are labelled accordingly. Nevertheless, because there could be legal subsequences belonging to a bigger illegal sequence, the architecture tries to label as many contexts as possible to provide that additional information. This functionality produces more understandable logs simplifying the process of pinpointing the cause of the infraction.

8 RELATED WORK

The problem of generating driving plans compliant with HoS regulations have already been addressed in (Goel, 2018) and (Goel and Irnich, 2017), and HTN planning can be considered an enabling technology for this problem. However, we are using this planning technique from a different perspective: recognizing a preexisting temporal plan. As previously shown, a key issue is to formalize the HoS regulation, and an alternative formalization can be found in (Goel, 2018), which is aimed to use classical scheduling techniques to check the compliance of schedules (only plan verification) or to generate compliant schedules (plan generation), but not suitable for the problem of plan-goal recognition here addressed.

This application can be, in part, viewed as an instance of Runtime Verification with Planning, where

Table 2: Labelling output after encountering an illegal sequence. The system adds as many contexts as possible to ease the interpretation of the sequence.

Original Log					Annotated Labels									
Driver	Start	End	Duration	Activity	Week	Day	DayType	Sequence	BreakType	Token	Legal			
driver1	11/01/2017 20:49	11/01/2017 20:50	1	Driving	2	10	unknown	unknown	uninterrupted	A	no			
driver1	11/01/2017 20:50	11/01/2017 20:52	2	Other						A	no			
driver1	11/01/2017 20:52	11/01/2017 23:06	134	Driving						A	no			
driver1	11/01/2017 23:06	12/01/2017 00:00	54	Break					B.T1	no				
driver1	12/01/2017 00:00	12/01/2017 03:00	180	Driving					unknown	A	no			
driver1	12/01/2017 03:00	12/01/2017 03:02	2	Other						A	no			
driver1	12/01/2017 03:02	12/01/2017 03:03	1	Driving						A	no			
driver1	12/01/2017 03:03	12/01/2017 03:06	3	Other						A	no			
driver1	12/01/2017 03:06	12/01/2017 03:13	7	Driving						A	no			
driver1	12/01/2017 03:13	12/01/2017 03:17	4	Other						A	no			
driver1	12/01/2017 03:17	12/01/2017 06:30	193	Break						unknown	no			
driver1	12/01/2017 06:30	12/01/2017 06:33	3	Driving						A	yes			
driver1	12/01/2017 06:33	14/01/2017 12:02	3209	Break						WR.T1	yes			
									ndd	unique	uninterrupted			

regulations are expressed in a suitable formal language, most often based on temporal logic, as described in (Bensalem et al., 2014). This paper is interestingly different, in that it encodes regulations involving metric temporal constraints, expressed in an attribute grammar. Our work is also related with conformance checking, in the area of Process Mining, that consists of determining whether an executed process conforms with a process model (Aalst, 2016). Nevertheless, we do not *only* provide an answer to whether a plan is or is not generated by a regulatory model, but additionally our approach classifies subsequences of the input plan and recognizes/identifies them with concepts of the model, providing an interpretation of the plan in such terms. Therefore, this is a clear instance of a Plan-Goal Recognition (PGR) problem that is traditionally addressed either by providing a *plan library* to represent the set of possible plans to be recognized, or by representing the behavior of the agent as a *planning domain* (also known as *PGR as Planning*) (Ramirez and Geffner, 2009).

9 CONCLUSIONS

We have presented a method intended to provide support to experts on the task of interpreting what drivers are or have been doing by recognizing their activity recorded in an event log according to HoS regulations. The main contribution is the proposal of a temporal HTN-based approach to classify and label the activities in a real log where the plans observed are temporal sequences of events, using an off-the-self temporal HTN planner, with a domain generated from an attribute grammar that describes de HoS regulation, configured to recognize and classify the events according to the goals represented as HTN tasks. The novelty here is that the planning domain requires to

represent temporal and numerical information, and up to authors' knowledge there is no approach to which fits that requirements. Another contribution is the formalization of concepts and constraints of the HoS regulation as an attribute grammar, a necessary step in the knowledge engineering process proposed to represent the HTN domain.

REFERENCES

- Aalst, W. v. d. (2016). *Process Mining: Data Science in Action*. Springer-Verlag, Berlin Heidelberg, 2 edition.
- Bensalem, S., Havelund, K., and Orlandini, A. (2014). Verification and validation meet planning and scheduling. *International Journal on Software Tools for Technology Transfer*, 16(1):1–12.
- Goel, A. (2018). Legal aspects in road transport optimization in europe. *Transportation research part E: logistics and transportation review*, 114:144–162.
- Goel, A. and Irnich, S. (2017). An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science*, 51(2):737–754.
- Goel, A. and Vidal, T. (2013). Hours of service regulations in road freight transport: An optimization-based international assessment. *Transportation science*, 48(3):391–412.
- Knuth, D. E. (1968). Semantics of context-free languages. *Mathematical systems theory*, 2(2):127–145.
- Meyer, C. M. (2011). European Legislation on Driving and Working Hours in Road Transportation. In Meyer, C. M., editor, *Vehicle Routing under Consideration of Driving and Working Hours: A Distributed Decision Making Perspective*, pages 9–24. Gabler, Wiesbaden.
- Ramirez, M. and Geffner, H. (2009). Plan recognition as planning. In *Proceedings of the 21st international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc, pages 1778–1783.