Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

Driver Activity Recognition by Means of Temporal HTN Planning*

Juan Fernandez-Olivares, Raul Perez
Department of Computer Science and Artificial Intelligence
University of Granada
faro@decsai.ugr.es, fgr@decsai.ugr.es

Abstract

When delivering a transport service, scheduled driver workplans have to be aligned with world wide complex hours of service (HoS) regulations which constraint the amount of working and driving time without resting. The activities of such workplans are recorded by onboard sensors in large temporal event logs. Transport companies are interested on recognizing what a driver is doing, based on the temporal observations from event logs, considering the terms defined by HoS regulations. This work presents an application of temporal HTN planning to plan and goal recognition that, starting from a real event log extracted from a tachograph, identifies different sub-sequences of a driver's daily and weekly driving activity and labels them according to the terms defined by HoS regulation.

Introduction

World wide transport authorities are imposing complex hours of service (HoS) regulations to drivers (Meyer 2011; Goel and Vidal 2013), which constraint the amount of working, driving and resting time when delivering a service. As a consequence, scheduled driving plans have to be aligned with laws that define the legal behavior of a driver.

A problem of paramount importance for transport companies is to determine whether driver activity conforms with HoS regulation to forestall illegal behaviour and avoid costs due to sanctions . Fortunately, the widespread adoption of onboard IoT devices in vehicle fleets enables recording the activity of drivers in event logs. Therefore, 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. It does not only consists on finding out drivers' workplan compliance of HoS regulation, known as plan verification (Barták, Maillard, and Cardoso 2018), but also on determining what a driver is doing, based on the observations from the event log, considering the behavior patterns defined by the HoS regulation. We are calling this problem *Driver Activity Recognition under HoS* and it is a problem with many similarities with plan and goal recognition (Simpson 2006) which normally is addressed by inferring which plan, from a known set of possible plans, an agent is executing based on observations of their actions (Geib and Goldman 2011). Particularly we are addressing this problem as Plan and Goal Recognition (PGR) as HTN planning (Höller et al. 2018) where, instead of a plan library or an operator-based model (Ramırez and Geffner 2009), an HTN domain is used as the model to describe the behavior rules of the agent. The main novelty in this work is that the observations of the driver are temporally annotated and, up to authors' knowledge current approaches for PGR as Planning do not address the representation of temporal and numerical information.

Therefore, in this paper we present an application that, starting from a real event log extracted from a tachograph, identifies different temporal sub-sequences of a driver's daily and weekly driving activity and labels them according to the terms defined by HoS regulation. The identification of temporal sub-sequences is an activity recognition problem that is addressed as a temporal HTN planning problem wherein (1) the domain describes the HoS regulation constraints as a hierarchy of tasks with temporal and numerical constraints, (2) the initial state represents a set of temporally annotated observations each one corresponding to an event of the original log, and (3) the goal-task is any of the top-most tasks of the domain hierarchy. Indeed, the temporal HTN problem can be seen as the high-level description of a parsing task to be performed over the event log considering the temporal HTN domain as a set of production rules of an attribute grammar. The result of this recognition process is a labeled event log that can be easily interpreted by company experts which can make more informed decisions considering the historic 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. Then, we briefly explain some background concepts needed to ease the overall description of the approach. We then present experimentation conducted over a proof of concept of the application, and discuss related and future work.

*This work has been partially supported by Spanish Government Projects MINECO RTI2018-098460-B-I00 and TIN2015-71618-R

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. Compesate with same duration before the end of the third week following the week considered.
WEEKLY REST	NORMAL	45hrs		

HoS CONCEPT	TYPE	ACCUMULATED DURATION		ADDITIONAL CONSTRAINS
		MIN	MAX	
BASIC SEQUENCE				Composed by any combination of [DOPI] whenever P= NO-BREAK
DRIVING PERIOD	CONTINUOUS	0	<=4.5hrs	Basic Sequence ended by a break_t1 or a Daily Rest.
DRIVING PERIOD	SPLITED	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.

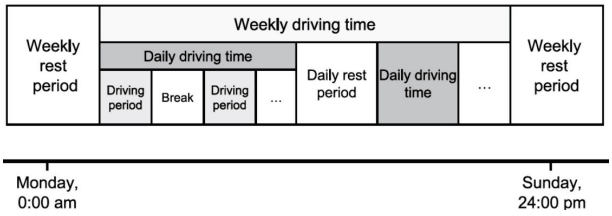


Figure 2: Hierarchical relations and structure of the different types of sub-sequences, extracted from (Meyer 2011).

Problem Description

We are collaborating with a company which provides decision support based on prediction services to its customers. They want to help them to govern the behavior of their drivers by predicting whether a driver is close to committing a regulation violation, or characterizing drivers according to their driving style with respect to the HoS regulation. They handed us an event log with thousands of events and asked us to label the records with the terms of the HoS regulation, so that an expert can directly interpret what a driver has been doing during the period of time encompassed by the event log.

Every event is a tuple $(a \ \ start \ \ end \ \ dur \ \ d\_id)$, where each component refers to: activity identifier, event start and end times, event duration, and driver identifier, 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. The semantics of each event is completed with the definitions 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).

Technical Challenges

According to previous description, the following *key features to be modeled* can be identified: (i) duration and temporal constraints over both actions/events and sequences of activities, (ii) numerical information and constraints, (iii) iteration constraints (maximum number of occurrences for an event), and (iv) hierarchical/compositional relationships between tasks at different levels of abstraction. Furthermore, *the problem consists of parsing the temporal event log* and *identifying which sub-sequences of events correspond with legal definitions of HoS regulations*. Therefore, the observed behavior has to be provided 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.

The main contribution of this work consists of the resolution of three technical challenges, related to both how the key features to be modeled are represented, and how the parsing of a temporal event log is faced: 1) *the recognition of plans involving temporal and numeric constraints* is addressed with a temporal planning approach; 2) since *plans have to be recognized following the HoS regulation*, we propose to use a temporal HTN process wherein the events are represented as PDDL primitive durative actions, the HoS high-level concepts are represented as high-level tasks, allowing the representation of recursive tasks; 3) in order to deal with *the formalization of the HoS regulation*, we propose a Knowledge Engineering process that starts from a formalization of the HoS regulation as an attribute grammar and translate it into a temporal HTN domain.

Background

Regarding knowledge representation and reasoning, we are using HPDL and SIADEX, a planning language and a planner designed to represent temporal HTN problems and successfully applied in serveral application domains (Fdez-Olivares et al. 2006; Fdez-Olivares et al. 2019). The syntax of HPDL embodies the syntax of PDDL 2.1 level 3 to represent primitive actions as durative actions, as well as other important PDDL concepts as derived literals, temporal predicates and temporal facts and timed initial literals in the initial state. Furthermore, it extends PDDL to represent tasks at different levels of abstraction and methods to describe alternative decomposition schemes for compound tasks (see Figure 3). HPDL provides additional features with respect to other HTN languages (Holler et al. 2019) which make it suitable for the purposes of this application:

1. Defining temporal constraints over the start and end point, as well as the duration of a task at any level. Any subaction in a method has three special variables `?start`, `?end` and `?dur` that represent it start and end time points as well as its duration, and some constraints on their value may be posted on them using a logical expression with relational operators (an example is shown in Figure 4).
2. Using inference tasks `(:inline <precondition> <effect>)` where `<precondition>` and `<effect>` are usual expressions for preconditions and effects of PDDL actions. They may be used as "on the fly" sub-tasks in a task decomposition scheme to infer new knowledge (asserting/retracting literals into the current state) or capturing information from the current world state.
3. Binding a variable with the evaluation of an expression, using the special predicate `(bind <var> <expression>)`.
4. Representing temporal facts, either as PDDL timed initial literals or facts embodying time data as for example `(start_action P15 "05/01/2017 15:46")`, a literal representing the start date of the action `P15`.

SIADEX is essentially a state-based forward HTN planner with the same foundations as SHOP (Nau et al. 2003), but extended with the management of temporal information at any level of the task hierarchy, besides the management of the temporal and numeric capabilities of PDDL 2.1 and the singular features of HPDL above described. A more detailed description of the planner can be found in (Castillo et al. 2006).

```
(:task <name>  :parameters <var-typed-list>
 (:method <name>
  :precondition <prec> ;;a PDDL logical expression
  :tasks ( ;;an (un)orderder list of one of ...
    ;; a "normal" task
    (<name> <var-typed-list>)
    ;; a task with temporal constraints
    (<temp_constraints> (<name> <var-typed-list>))
    ;; an inference task
    (:inline <condition>  <consequence>)))
 (:method ...)
 ...
 )
```

Figure 3: An HPDL task is represented with a name, a list of typed parameters and a set of decomposition methods. A decomposition scheme is a (partially) ordered list of subtasks, where every element can be of one of these three types: normal, temporally constrained or inference tasks.

Application Overall Description

The methodology we have followed to provide a solution to the problem of driver activity recognition under HoS regulation consists of the following steps:

1. Generating a set of temporal observations represented in PDDL from an event log, which are part of the initial state of an HTN problem.
2. Representing the recognition of an event as a temporal HTN problem.
3. Formalizing HoS rules as productions of an attribute grammar.
4. Translating the grammar into a temporal HTN domain, aimed at representing the parsing of the event log as an HTN problem where (1) terminals are recognized as temporal events and (2) nonterminals are recognized according to grammar rules.
5. Extending that domain to be capable of both recognizing and labeling the events for the event log to be easily interpretable.

The following sections describe in detail these steps.

Generating HPDL 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. Every observation is in turn represented with a set of classical and temporal PDDL facts. 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 "05/01/2017 15:46" "05/01/2017 16:54" 68
driver1)
PDDL-facts:
(is_action P15) ;; P15 is an action identifier
(is_typeP P15)  ;; P15 is of type "P"
(index_action P15 39)  ;;P15 is the 39th observation
(start_action P15 "05/01/2017 15:46") ;;start of P15
(end_action P15 "05/01/2017 16:54")   ;;end of P15
(duration_action P15 68)              ;;duration of P15
(parameters_typeP P15 driver1) ;;parameters of P15
```

Additionally we have defined the PDDL function `(current_index_action)` which keeps track of the index of the current observation, initially set to 0 in the initial state. This function is needed to represent a "virtual" pointer to the current event/observation being recognized.

Event Recognition with HPDL

Each $type$ of event $[Driving, Other, Pause, Idle]$ has an associated primitive durative action $\langle type \rangle$_p in the domain, that represents the planner has processed (read) an event of that type. Therefore, recognizing an concrete event that is at any given position k with label $type$ consists of adding its corresponding primitive action to a plan 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. The condition (ii) is checked by the planning process itself, but (i) has to be hand coded in the HTN domain. In the following we show an illustrative example.

Figure 4 shows the compound task `recognize_PAUSE`, which is applied when the reading pointer of the event log is pointing at an event of type $Pause$, and it is used to (1) add to the plan a durative primitive action `(PAUSE_p ?driver ?dur)` representing that a Pause has been read by the planner, and (2) bind the value of its argument `?dur` to the duration observed. This is clarified below.

The derived predicate is used to determine the type of the observation the reading pointer index is pointing at, and it is used in the precondition of the method to capture the value of the index invariable `?k` and the type of observation in variable `?sa`. The precondition of the first :inline task is used to capture the value of parameters and temporal information about the observation pointed by the current index, using the corresponding predicates used to represent a temporal observation. The primitive task represents the "token" (a Pause) to be read and the method describes in essence that a primitive task of type $Pause$ has to be added to the plan

```
(:durative-action PAUSE_p
   :parameters (?c - Driver ?dur - number)
   :duration (= ?duration ?dur)
   :condition ()
   :effect ()
)

(:derived (currentindex_is_typeP ?k ?sa)
   (and (bind ?k (current_index_action))
        (index_action ?sa ?k)
        (is_typeP ?sa))))

(:task recognize_PAUSE
   :parameters(?c - Driver ?dur - number)
    (:method type_PAUSE
        :precondition (currentindex_is_typeP ?k ?a)
        :tasks (
          (:inline (and (parameters_typeP ?a
           ?driver)
             (start_action ?a ?begin)
             (end_action ?a ?final)
             (duration_action ?a ?dur)) ())
          ((and (= ?start ?begin)
                (= ?end ?final)
                (= ?duration ?dur))
             (PAUSE_p ?driver ?dur))
          (:inline ()
           (increase (current_index_action) 1)))
```

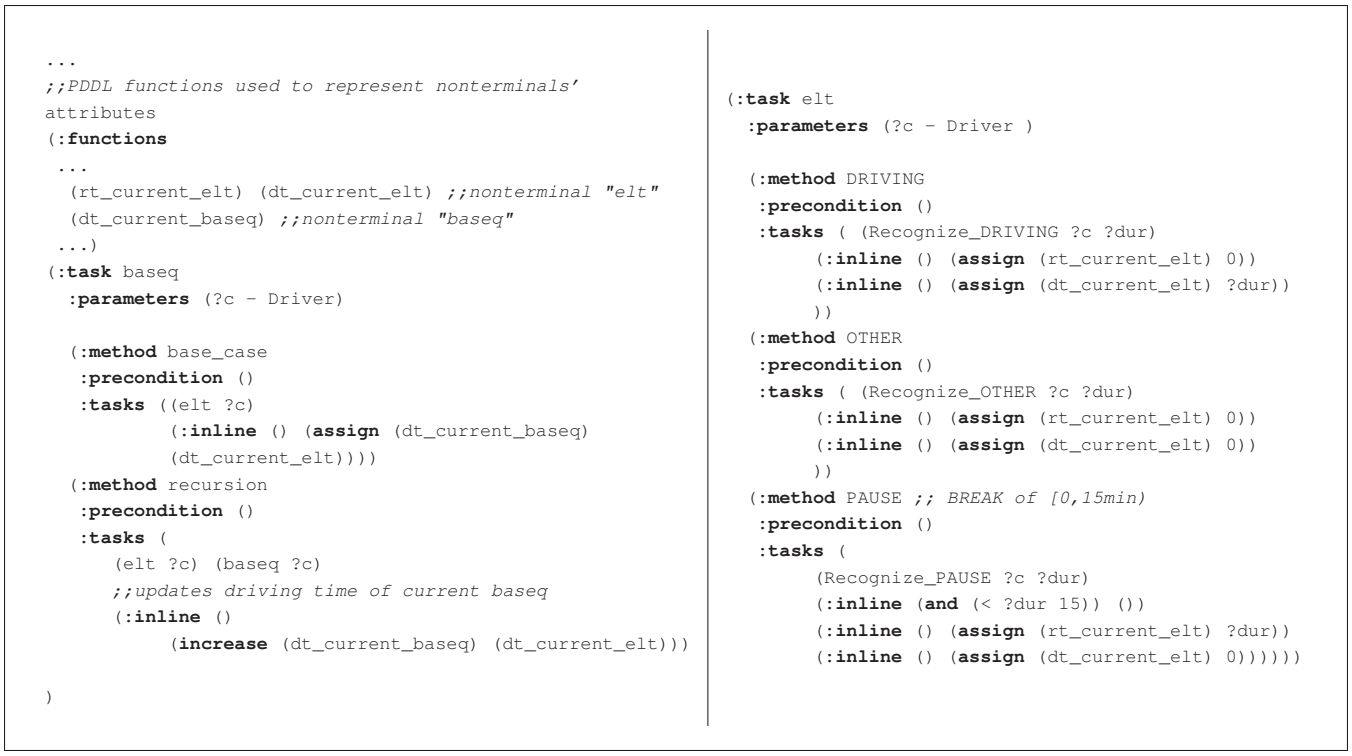
Figure 4: Recognizing an event of type $Pause$ with HPDL. Singular features of HPDL are framed.

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 which, in turn, is captured by the first :inline task. This is the point where temporal planning capabilities play an central role since the method will fail if the temporal constraints cannot be met. If the primitive task is succesfully inserted the index moves to the next position.

Recognizing HoS Concepts with HPDL

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 temporal and numeric constraints with grammar attributes. An attribute grammar extends productions with *semantic rules*[1] of the form $\{\langle condition \rangle \langle assignments \rangle\}$,

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



```
...
;;PDDL functions used to represent nonterminals'
attributes
(:functions
 ...
  (rt_current_elt) (dt_current_elt) ;;nonterminal "elt"
  (dt_current_baseq) ;;nonterminal "baseq"
 ...)
(:task baseq
  :parameters (?c - Driver)

  (:method base_case
   :precondition ()
   :tasks ((elt ?c)
           (:inline () (assign (dt_current_baseq)
           (dt_current_elt))))
  (:method recursion
   :precondition ()
   :tasks (
      (elt ?c) (baseq ?c)
      ;;updates driving time of current baseq
      (:inline ()
           (increase (dt_current_baseq) (dt_current_elt)))

)
```

```
(:task elt
  :parameters (?c - Driver )

  (:method DRIVING
   :precondition ()
   :tasks ( (Recognize_DRIVING ?c ?dur)
        (:inline () (assign (rt_current_elt) 0))
        (:inline () (assign (dt_current_elt) ?dur))
        ))
  (:method OTHER
   :precondition ()
   :tasks ( (Recognize_OTHER ?c ?dur)
        (:inline () (assign (rt_current_elt) 0))
        (:inline () (assign (dt_current_elt) 0))
        ))
  (:method PAUSE ;; BREAK of [0,15min)
   :precondition ()
   :tasks (
        (Recognize_PAUSE ?c ?dur)
        (:inline (and (< ?dur 15)) ())
        (:inline () (assign (rt_current_elt) ?dur))
        (:inline () (assign (dt_current_elt) 0))))))
```

Figure 5: HPDL tasks for the recognition of a basic sequence of events as described by the HoS regulation. The task `elt` recognizes any of the possible events that can be found in an event log, and the task `baseq` is a recursive definition of the recognition of a sequence of elements `elt`.

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), as well as the methodology used to translate such productions into HPDL tasks.

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. Every non-terminal has associated two attributes that represent the *driving time* (attribute `dt`) and *pausing time* (attribute `rt`) of the current recognized token [2].

Breaks recognition. The pausing time of any pause is further used to discriminate between different types of Breaks or Rests, according to rules in Figure 1 (left-hand table).

```
b_t1 : P {(and (P.dur >= 45min)(P.dur < 3hrs)
          (b_t1.rt := P.dur)
b_t2 : P {(and (P.dur >= 15min)(P.dur < 30min))
          (b_t2.rt := P.dur)}
b_t3 : P {(and (P.dur >= 30min)(P.dur < 45min))
          (b_t3.rt := P.dur)}
```

[2]Grammar terms attributes are represented with the "dot" notation used to refer the items of a structure.

Rests recognition. Daily and weekly rests (`rd` and `rw`) may be normal or reduced, and are defined according to the duration of pauses described in Figure 1 (left-hand table).

```
rd :         rd_normal {rd.rt := rd_normal.rt} |
             rd_reduced {rd.rt := rd_reduced.rt};
rd_normal :  P {(and (P.dur >= 11h) (P.dur < 24h)
                (rd_normal.rt := P.dur) };
rd_reduced : P {(and (P.dur >= 3h) (P.dur < 11h)
                (rd_reduced.rt := P.dur)};
rw :         rw_normal {rw.rt := rw_normal.rt} |
             rw_reduced {rw.rt := rw_reduced.rt};
;;rw_normal and rw_reduced are defined similarly.
```

Basic sequences recognition. The following production defines the elements of a basic HoS sequence as described in Figure 1 (right-hand table).

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

The HoS regulation establishes that a basic sequence is a sequence of any number of activities such that the duration of any $Pause$ is strictly less than 15 mins. The following production represents that rule and, moreover, 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)}
         elt baseq2 {(baseq.dt := elt.dt+baseq2.dt)};
```

Representing tasks from productions. Figure 5 shows as an example the HPDL representation of the production rules for `elt` and `baseq`. The tasks described in this example are the result of a knowledge engineering process that represents every recognition task as an HPDL task, from a formalization of it as an attribute grammar production rule. In this process, every production $\langle lhs \rangle : \langle rhs_1 \rangle | ... | \langle rhs_n \rangle$ of the grammar is translated (manually) into a recognition task for the symbol lhs with name lhs, typed parameters list `(?c - Driver)`, and n methods, each one corresponding to a rhs_i. For every method the following steps are performed:

- The method precondition is empty.
- Grammar attributes are represented as PDDL functions. Concretely, every attribute a of a nonterminal t is represented as a PDDL function $(\langle t \rangle\_\langle a \rangle)$. For example `elt.dt` is represented as the function `(dt_current_elt)` used to store the value of the driving time of the current `elt` processed.
- Assignments of attributes are represented as HPDL (or PDDL) assignment operations in the effects of inference tasks. For example `baseq.dt := elt.dt` is represented as `(:inline () (assign (dt_current_baseq) (dt_current_elt)))`.
- Terminal and nonterminal symbols are represented by their corresponding recognizing tasks.
- Regarding conditions in production rules, we are assuming that those conditions which refer to a symbol's attribute in the rhs of productions are evaluated after the symbol has been processed. For example, testing the value of the duration of a $Pause$ in `(P.dur < 15min)` is evaluated after a symbol `P` is processed. Therefore, in order to keep this order in the evaluation process, conditions are represented as the precondition of an inference task, and the task is *ordered after the subtask* which recognizes the symbol the condition refers to. For example, as shown in Figure 4 the inference task `(:inline (and (< ?dur 15)) ())` is ordered after the task to recognize a $Pause$ which, in turn, captures the value of the variable `?dur`.

Finally, Figure 6 shows the production rules to recognize the rest of HoS concepts described in Figure 1. The non terminal symbols of the right-hand-side of every production are used as the labels of the event labeling process, performed over the entire event log, briefly described in next section.

Labeling Events with HPDL

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 four labels[3] according to the contexts *day* (with possible values `ndd` or `edd`), *driving period* (`continuous` or `splitted`), *sequence* (`cdds` or `cdde`) and *token* (`elt`, `b_t1`, `b_t2`, `b_t3`, `rd_normal`, `rd_reduced`, `wr_normal`, `wr_reduced`).

We have made extensive use of the features of HPDL (and the control search mechanism of the planner) to write an HTN domain, which allows to label records of the event log, as an extension of the HTN domain defined in last sections. Concretely, every task in the right hand side of a production $\{$`a : b1 .. bn` $\}$, is extended as $\{$`a : sb1 b1 eb1 ... sbn bn ebn`$\}$ where the new tasks sb_i,eb_i are used to (1) delimit the start (sb_i) and end (eb_i) of the decomposition of a task (sb_i), (2) add to the state information about the label corresponding to that task (which can be interpreted as the current decomposition context). For example the production $\{$`cdd : cdds cdde`$\}$ is extended as $\{$`cdd : scdds cdds ecdds scdde cdde ecdde`$\}$, `scdds` asserts the context `'cdds'` to the state, and `ecdds` retracts that context. Finallly, the primitive durative actions associated with each event are extended with new parameters (one for each type of label) and tasks devoted to recognize events (as the one shown in Figure 4) are extended to capture the information added to the state by the newly added delimiting tasks, and adding primitive tasks which embody information about the context.

Experimentation

We have conducted an experimentation in order to validate a proof of concept implementation where the events of an initial event log have been annotated with labels corresponding to the nonterminal symbols defined in previous section. Figure 7 shows an example of the original and annotated event logs. In the experiments we have generated an initial state that contains 200 observations extracted from the event log plus the necessary symbol definitions to manage all the possible types of actions. Moreover, the initial state also contains the initialization of the PDDL functions used to represent the different counters and accumulators to store the values of driving and resting time. The goal consists of the task `(wdd driver1)` what means that it is required to recognize the weekly activity of `driver1` in the event log. The temporal HTN domain represents the recognizing and labeling tasks as explained in the previous sections and the planner carries out a planning process that involves recognizing and labeling the events which encompasses one week. As can be seen in Figure 7, the original event log can be interpreted by using the labels of its annotated version. In this example, every activity (or event) is annotated at token level with its corresponding label (an expert can interpret directly with type of Pause is being carried out by the driver). And, what is much more interesting, if we put together the information at sequence, driving period and day level, in this case the system informs that the driver, during a day, has carried out a normal daily driving, composed of a starting sequence of a splitted driving period and an ending sequence of continuous driving period.

Once the post-processed event log was shown to users, they provided us an evaluation and feedback extremely positive, since they do not have at present an automated means

[3] We are assuming four labels and a reduced set of possible values for each label to ease the explanation.

```
;;a weekly driving sequence is five daily driving
sequences and an ending day
wd : (dd)^5 end_day
     ({dd_1.dt +..+ dd_5.dt + end_day.dt <= 56)};

;;a daily driving sequence can be normal or extded.
dd : ndd {dd.dt:=ndd.dt} |
     edd {dd.dt:=edd.dt};

;;normal daily driving is a combination of driving
periods with less than 9hrs of driving
ndd : cdd {(cdd.dt<= 9hr)
           ndd.dt:=cdd.dt};

;extended daily driving allows upto 10hrs of driving
edd : cdd {(and (cdd.dt > 9h)
                (cdd.dt<= 10hr))
           edd.dt:=cdd.dt};
```

```
;; a combination of driving periods has a start and an end
cdd : cdds cdde
      {cdd.dt:= cdds.dt + cdde.dt};
end_day : cdds cdde_w
          {end_day.dt := cdds.dt + cdde_w.dt}
;;the start of a combination is a continuous driving period
;;or a splitted driving period
cdds : seq b_t1 {cdds.dt:=seq.dt} |
       seq b_t2 seq_2 b_t3
       {cdds.dt:=seq.dt+seq_2.dt};
;;the end of a combinantion of driving period is the same
;;but ending in a daily rest
cdde : seq rd {cdde.dt:=seq.dt} |
       seq b_t2 seq_2 rd
       {cdde.dt:=seq.dt+seq_2.dt};
;;similarly for the end of a week
cdde_w : seq rw {cdde_w.dt:=seq.dt} |
         seq b_t2 seq_2 rw
         {cdde_w.dt:=seq.dt+seq_2.dt};
```

Figure 6: Production rules to recognize the remaining elements of HoS regulation. All these rules can be translated into HPDL, however we only show the grammar productions for the sake of clarity and generality.

to get a descriptive model easily interpretable for large event logs. Experts have not found a commercial tool that covers their need, and up to authors' knowledge, there are no other approaches that address the problem of recognizing driver activity under HoS regulation.

Related Work

HTN planning could be seen as a very suitable technique to solve the problem of *generating* driving plans compliant with HoS regulations. In fact, other approaches (which do not use HTN planning) have been developed to solve route planning problems under HoS regulations minimizing transportation costs (Goel 2018)(Goel and Irnich 2017). Although addressing this problem with temporal HTN planning would be of great interest, we are using this planning technique from a different perspective: recognizing a previously synthesized 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 regulations are expressed in a suitable formal language, most often based on temporal logic, as described in (Bensalem, Havelund, and Orlandini 2014). This paper is interestingly different, in that it encodes regulations involving metric temporal constraints, expressed in an attribute grammar. (Barták, Maillard, and Cardoso 2018) addresses the problem of (non temporal) plan verification, integrating HTN domains (with no temporal or numeric information) with attribute grammars. 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 (in our case the HoS regulation represented as a temporal HTN domain), 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*) (Ramırez and Geffner 2009).

Concretely, we follow a *PGR as Planning* approach since we are providing as input a (temporal HTN) planning domain. In order to compare with other approaches, it is worthy to note that we are assuming full observability, noiseless and complete event sequences, no ambiguity, and an offline recognition process. Most approaches to plan and goal recognition are focused on the offline analysis, and very few address an online approach focused in path planning problems (Masters and Sardina 2019) or human-robot interaction (Levine and Williams 2014; Freedman and Zilberstein 2017) . A large bunch of work addresses ambiguity and uncertainty in the field of plan recognition (Ramírez and Geffner 2010; Sohrabi, Riabov, and Udrea 2016) even for a goal recognition process supported by HTNs (Blaylock and Allen 2006). These works are motivated by the existence of noisy sensors or by incompleteness in the behavior model. This is not our case, since the events provided by digital tachographs are noiseless, and the HoS regulation is a complete and pre-

ORIGINAL EVENT LOG						ANNOTATED EVENT LOG			
						Labels			
Driver	Start		End	Duration	Activity	day	driving period	sequence	token
driver1	4/01/17	09:25	09:29	0h 04m	Driving	ndd	splitted	cdds	elt
driver1	4/01/17	09:29	09:47	0h 18m	Pause	ndd	splitted	cdds	B_T2
driver1	4/01/17	09:47	09:50	0h 03m	Driving	ndd	splitted	cdds	elt
driver1	4/01/17	09:50	09:54	0h 04m	Other	ndd	splitted	cdds	elt
driver1	4/01/17	09:54	11:07	1h 13m	Pause	ndd	splitted	cdds	B_T3
driver1	4/01/17	11:07	15:22	4h 15m	Driving	ndd	splitted	cdds	elt
driver1	4/01/17	15:22	15:47	0h 25m	Pause	ndd	splitted	cdds	B_T2
driver1	4/01/17	15:47	15:50	0h 03m	Driving	ndd	splitted	cdds	elt
driver1	4/01/17	15:50	16:49	0h 59m	Pause	ndd	splitted	cdds	B_T3
driver1	4/01/17	16:49	21:11	4h 22m	Driving	ndd	continuous	cdde	elt
driver1	4/01/17	21:11	21:13	0h 02m	Other	ndd	continuous	cdde	elt
driver1	4/01/17	21:13	06:21	9h 08m	Pause	ndd	continuous	cdde	rd_red

Figure 7: An excerpt of the event log annotated with labels easing the interpretation of driver activity.

cise set of rules with no room for unforeseen behaviors. Hence, at this moment, we do not need to address uncertainty on goal recognition, thus a deterministic approach is much more suitable. We have observed in some cases incompleteness, but it can be straightforwardly addressed with the temporal process of the planner. For example, once detected a 'gap' in the event log, it is possible to insert directly an idle or pause activity with a duration equal to the size of the gap.

Regarding temporal plan recognition, few works address problems where observations have temporal information, and, up to authors' knowledge, no one addresses the problem using a temporal planning domain. (Geib and Goldman 2009) proposes PHATT, which uses temporal constraints and variable bindings, thus going beyond the recognition of propositional activity streams. However, the use of a temporal model is only mentioned as a suggestion on how to improve PHATT with an external STP solver for temporal reasoning. The approach in (Levine and Williams 2014) assumes fully observable, noiseless and complete event streams, as well as no ambiguity (as ours), and it recognizes temporal plans online (ours is offline). However, they require a plan library represented with TPNUs (Temporal Plan Networks with Uncertainty) and use *ad hoc* plan recognition algorithms, whereas, we use an off-the-shelf temporal HTN planner instead. Further, TPNUs lack of key aspects to represent HoS regulations like hierarchical compositional relationships, (bounded) recursion to represent iteration constraints, or representing and reasoning with numerical information. Indeed, as stated in (Geib and Steedman 2007), the knowledge representation models used for plan recognition based on plan libraries can generally be seen as special cases of HTNs. Moreover, those based on planning domains use either strips-based action models (Ramírez and Geffner 2010; Sohrabi, Riabov, and Udrea 2016) or non temporal HTN (Höller et al. 2018), thus lacking of enough expressiveness to describe HoS regulations. This is the main aspect that made us to build a new proposal based on temporal HTN to solve the problem of driver activity recognition under HoS regulations.

Conclusions

We have presented a novel planning application that brings the worlds of Data Analytics, IoT and Automated Planning and Scheduling together. The approach provides support to experts on the task of interpreting what drivers are or have been doing by recognizing their activity recorded in an event log. The main contribution is the proposal of a temporal HTN-based approach to plan and goal recognition where the plans observed are temporal sequences of events, using an off-the-self temporal HTN planner, with a domain configured to recognize and classify the events according to the goals represented as HTN tasks. The novelty here is that, assuming that we are clearly not using a plan library, but a planning domain as the model of behavior, the planning domain requires to represent temporal and numerical information, and up to authors' knowledge there is no approach to PGR as Planning 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.

Regarding future work, it is worth noting that the main interest, and the ultimate goal of the company is to build an intelligent assistant to provide decision support services to both drivers and companies' decision makers. This is a research direction aligned with the concept of *assistive interaction* (Freedman and Zilberstein 2017), that advocates for the integration of plan recognition and planning. In this way, the recognition of driver's intent is a previous stage needed to respond with a generated plan adapted to the currently recognized task. Precisely, one of the main advantages of using a temporal HTN planner to carry out a recognition process as the one here presented, is that it paves the way to generate regulation-compliant plans adapted to the intent of the driver. Therefore, this work is the first step and part of a more ambitious project, proposed by the company, to

develop a predictive and prescriptive model able to detect whether a driver is close to have an illegal behavior, and provide a response by adapting the driving plan to the HoS regulation.

References

Aalst, W. v. d. 2016. *Process Mining: Data Science in Action*. Berlin Heidelberg: Springer-Verlag, 2 edition.

Barták, R.; Maillard, A.; and Cardoso, R. C. 2018. Validation of hierarchical plans via parsing of attribute grammars. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

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.

Blaylock, N., and Allen, J. 2006. Fast hierarchical goal schema recognition. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, 796. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Castillo, L. A.; Fernández-Olivares, J.; García-Pérez, Ó.; and Palao, F. 2006. Efficiently handling temporal knowledge in an htn planner. In *ICAPS*, 63–72.

Fdez-Olivares, J.; Castillo, L.; Garcıa-Pérez, O.; and Palao, F. 2006. Bringing users and planning technology together. experiences in siadex. In *Proc. ICAPS*, 11–20.

Fdez-Olivares, J.; Onaindia, E.; Castillo, L. A.; Jordán, J.; and Cózar, J. A. 2019. Personalized conciliation of clinical guidelines for comorbid patients through multi-agent planning. *Artif. Intell. Medicine* 96:167–186.

Freedman, R. G., and Zilberstein, S. 2017. Integration of planning with recognition for responsive interaction using classical planners. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Geib, C. W., and Goldman, R. P. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence* 173(11):1101–1132.

Geib, C. W., and Goldman, R. P. 2011. Recognizing plans with loops represented in a lexicalized grammar. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 958–963.

Geib, C. W., and Steedman, M. 2007. On natural language processing and plan recognition. In *IJCAI*, volume 2007, 1612–1617.

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.

Goel, A. 2018. Legal aspects in road transport optimization in europe. *Transportation research part E: logistics and transportation review* 114:144–162.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2018. Plan and Goal Recognition as HTN Planning. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 466–473.

Holler, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2019. HDDL - A Language to Describe Hierarchical Planning Problems. In *ICAPS 2019 Workshop on Hierarchical Planning*.

Knuth, D. E. 1968. Semantics of context-free languages. *Mathematical systems theory* 2(2):127–145.

Levine, S. J., and Williams, B. C. 2014. Concurrent plan recognition and execution for human-robot teams. In *Proceedings of the Twenty-Fourth International Conferenc on International Conference on Automated Planning and Scheduling*, 490–498. AAAI Press.

Masters, P., and Sardina, S. 2019. Cost-Based Goal Recognition in Navigational Domains. *Journal of Artificial Intelligence Research* 64:197–242.

Meyer, C. M. 2011. European Legislation on Driving and Working Hours in Road Transportation. In Meyer, C. M., ed., *Vehicle Routing under Consideration of Driving and Working Hours: A Distributed Decision Making Perspective*. Wiesbaden: Gabler. 9–24.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. Shop2: An htn planning system. *Journal of artificial intelligence research* 20:379–404.

Ramırez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc*, 1778–1783.

Ramírez, M., and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 1121–1126.

Simpson, R. 2006. Plans and Planning in Smart Homes. In *Designing Smart Homes*, volume 4008. Berlin, Heidelberg: Springer Berlin Heidelberg. 71–84.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *IJCAI*, 3258–3264.