Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 127 – 136

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

Using an improved rule match algorithm in an expert system to detect broken driving rules for an energy-efficiency and safety relevant driving system

Emre Yay[a]*, Natividad Martínez Madrid[a], Juan Antonio Ortega Ramírez[b]

[a]Reutlingen University, School of Informatics, Alteburgstr. 150, Reutlingen 72762, Germany
[b]*Universidad de Sevilla, Dpto. Lenguajes y Sistemas Informáticos, Av. Reina Mercedes s/n., Sevilla 41012, Spain*

Abstract

Vehicles have been so far improved in terms of energy-efficiency and safety mainly by optimising the engine and the power train. However, there are opportunities to increase energy-efficiency and safety by adapting the individual driving behaviour in the given driving situation. In this paper, an improved rule match algorithm is introduced, which is used in the expert system of a human-centred driving system. The goal of the driving system is to optimise the driving behaviour in terms of energy-efficiency and safety by giving recommendations to the driver. The improved rule match algorithm checks the incoming information against the driving rules to recognise any breakings of a driving rule. The needed information is obtained by monitoring the driver, the current driving situation as well as the car, using in-vehicle sensors and serial-bus systems. On the basis of the detected broken driving rules, the expert system will create individual recommendations in terms of energy-efficiency and safety, which will allow eliminating bad driving habits, while considering the driver needs.

Peer-review under responsibility of KES International.

Keywords: Driving System, Expert Sytem, Rule Match Algorithm, Rete Algorithm, Energy-Efficiency, Safety, Behavioural Computing, Intelligent System, Adaptive Driving System

* Corresponding author. Tel.: +49-7121-271-4063; fax: +49-7121-271-90-4063.
E-mail address: emre.yay@reutlingen-university.de.

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.092

1. Introduction

Saving energy and protecting the environment have become fundamental for politics and society1. Furthermore, as a result of the increasing number of cars and drivers, more accidents and fatalities on the roads have been determined2. The driving behaviour has a great impact on safety3 and on the energy consumption of a vehicle. Thus, the adaptation of the driving behaviour can save energy up to 30%4,5,6 and increase the road safety.

Considering the facts above a driving system is introduced in this paper, which is currently under development. Its goal is to optimise the driving behaviour in terms of energy-efficiency and safety by giving adequate recommendations for the current driving situation. The recommendations depend on the chosen area of improvement like energy-efficiency and/or safety. If the driver adheres the given recommendations it is possible to fulfil the energy-efficiency and safety potential of adapted driving.

There are already several driving systems, whose goal is to optimise the driving behaviour by giving energy-efficiency or safety relevant hints7,8. However these driving systems cover either the area of energy-efficiency or safety. In contrast, the guiding system introduced in this paper will try to improve both areas. Moreover, the typical driving behaviour will be represented using a driving profile, which allows the adaptation of the guiding system to the individual driving behaviour. This makes it possible to create recommendations based on any negative change of the driving behaviour or the driver condition. Furthermore, the acceptance of the driving system could be increased as only useful recommendations will be shown to the driver. The recommendations will be given on time, as the driving system predicts the state of the car. Thus, the reaction of the driver to the dangerous driving situation will be appropriate. The first prototype of the driving system is developed on the basis of a driving simulator. The second prototype will be connected to a real car, to test the driving system in a real environment.

The following chapter gives a brief overview over energy-efficient or safety relevant driving systems. Chapter 3 explains the architecture of the driving system. The rule match algorithm, used in the expert system, is introduced and evaluated in the Chapters 4 and 5 respectively. Finally, a conclusion of this work and a forecast about the future work is given in the Chapter 6.

2. Related work

An energy-efficient and safe driving behaviour is described by a set of driving rules. As these rules have to be adhered to achieve the goal to save energy and to increase the safety, the cooperation of the driver is needed.

Van Mierlo et al.6 evaluated several energy-efficient related driving rules. The results showed a decrease of the energy consumption and vehicle emissions, when the drivers interpreted the rules correctly. Furthermore, the driving speed decreased during the practice of the driving rules. According to Haworth et al.4 a reduced driving speed leads to an increase of the road safety.

“ANESA”9 is a driving system trying to reduce the energy consumption of the car through freewheeling. This approach saves up to 13%, when the drivers followed the instructions of the driving system on time. However, freewheeling is only one aspect of energy-efficient driving, therefore the energy-savings could be more increased using the driving rules mentioned in “Driving style and traffic measures”6.

Another driving system[10] is based on the interaction between the mobile phone and the car. Its focus is the education of the driver in eco-driving by giving advices to eliminate bad driving habits. The driving system runs on a mobile device, because the needed information is gathered through the diagnostic port of the car and the internet connection of the device. The internet connection of the mobile device may not be available during the whole journey, therefore it is not guaranteed that the driving system is able to obtain all needed data for further processing. Furthermore, the driving system does not consider the individual driving behaviour, which can be used for the adaptation of the driving system to the driver.

Car manufacturers also research on energy-efficient and ecological driving systems, for instance the driving system of Kia7. It gives feedback to the driver using two different coloured lamps, which mean energy-efficient driving and stand-by of the driving system or normal fuel consumption of the car. Furthermore, the driving system shows neither the wrongdoings of the driver nor gives driving hints to the driver. This would allow eliminating bad driving habits, which are the main cause of inefficient driving behaviour.

An energy-efficient driving behaviour has also positive effects on safety, as it prevents aggressive driving, which is the main cause of accidents4.

Besides the driving systems with respect to energy-efficiency, there are also safety relevant driving systems like "DAISY"[11]. It monitors the driver, the current driving situation and the driver condition to create warnings in dangerous situations, especially in situations that are susceptible for distractions. However, "DAISY" does not try optimising the driving behaviour although bad driving habits of the driver might have caused the dangerous situation.

Another driving system with the focus on safety is called "DriveDiagnostics"8. It has the goal to educate the driver in terms of safety. Therefore, "DriveDiagnostics" monitors and analyses the car movement to indicate the trip safety. The driving system has a real time feedback, which warns the driver when his current driving behaviour does not match his typical driving behaviour or when the driver drives aggressively. In contrast, the offline feedback of "DriveDiagnostics" shows the average trip safety to the driver after the journey based on the recorded data during the journey. The safety could be increased more by observing additionally the driver condition, which would allow the recognition of an uncommon driver condition like fatigue using tracking systems[12] and drowsiness using vital sensors[13]. Thus, his/her condition could be additionally the basis for detecting dangerous situations.

The driving systems presented in this chapter cover either the area of energy-efficiency or safety. They also do not consider the individual driving behaviour or the driver condition, which are also important factors to improve energy-efficiency and safety. In contrast, the driving system introduced in this paper adapts itself to the individual driving behaviour as well as considers the driver condition. Moreover, it covers both areas: energy-efficiency and safety. This allows the creation of individual driving hints in terms of energy-efficiency and safety, while considering the driver needs.

3. Architecture

The driving system presented in this paper is based on a multi-tier architecture, which is shown in Fig. 1. The driving system has three main components, which will be described in the following:

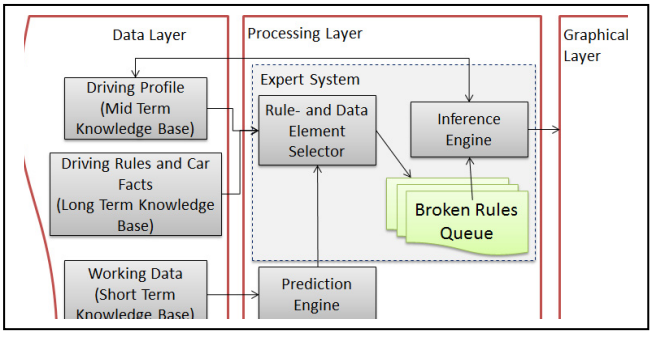

Fig. 1. An excerpt of the driving system architecture. The figure shows only modules that are important for the expert system.

- Data Layer: It is responsible for gathering all relevant data from the car, the driver and the environment. Therefore, it is connected to the in-car serial-bus systems to receive the information from the car, to vital sensors for monitoring the driver and to other sensors, which are relevant for retrieving information about the environment, like the weather condition. The incoming data is then aggregated using fuzzy logic 14. Based on the incoming and aggregated data, a driving profile is generated, which describes the typical driving behaviour of the driver. Beside these tasks, the "Data Layer" administrates and stores all relevant information, which is needed for further processing.
- Processing Layer: The information, stored and administrated in the "Data Layer", is used in the "Processing Layer" to analyse the driving behaviour. First, it predicts the state of the car using the stored data in the "Data Layer". On the basis of the prediction and the analysis of the driving behaviour, it generates individual recommendations, which guide the driver to drive energy-efficient or safe.
- Graphical Layer: Its main purpose is the rendering of the graphical user interface on the in-vehicle display unit. It also shows the created driving hints to the driver using for example the graphical user interface or an acoustic signal. Furthermore, it provides the opportunity to configure the driving profile by choosing the areas, which the driver wants to be improved like energy-efficient, safety or both areas.

In the following the Data Layer is briefly described. However, as the main focus of this paper is the process of the recommendation creation, the "Processing Layer" will be described in detail.

3.1. Data layer

The "Data Layer" is connected to the in-vehicle serial-bus systems and to different sensors to retrieve information about the car, the driver and the environment. In the first prototype the CAN (Controller Area Network) serial-bus system is used to get the information about the car, like driving or engine speed. Additionally, an ear sensor monitors the pulse of the driver, which allows the calculation of the heart-rate coherence. This value can be used to indicate drowsiness[15] and stress[16] of the driver. As the environment of the car has also an influence on energy-efficient and safe driving, for example the weather condition, different sensors are connected to the "Data Layer" for measuring the influences of the environment.

Based on the incoming data, a driving profile of the driver is generated, which represents his/her typical driving behaviour. The driving profile is stored in the "Mid Term Knowledge Base". However, during the initialisation of the driving profile, it has to be updated until it has enough data to represent the typical driving behaviour. Besides the raw incoming data, the driving profile considers also aggregated data during the profile update process. The aggregation is done using fuzzy logic, as some data has more value when it is fuzzy. The aggregated is stored along with the incoming data in the working memory, which is placed "Short Term Knowledge Base". Finally, the "Long Term Knowledge Base" stores information about the car, like mileage, and the driving rules.

3.2. Processing layer

The "Processing Layer" analyses the driving behaviour with respect to energy-efficiency and safety rules and generates recommendations, based on the results of the analysis, the individual driving behaviour and the reaction of the driver to the given recommendation. For these tasks the "Processing Layer" is using an expert system, which is separated in two modules: the "Rule- and Data Element Selector" and the "Inference Engine". Moreover, the driving system tries to give an early feedback to the driver, by predicting the state of the car in the "Prediction Engine" using the ARMA (Autoregressive-Moving Average) algorithm[17]. This allows giving recommendations on time, so that the driver has enough time to avoid the breaking of the driving rule. On the basis of the prediction, the expert system is able to generate recommendations before a breaking of a driving

rule occurs. The data, which is used in the "Processing Layer", is received from the "Data Layer", respectively from the "Short-", "Mid-" and "Long Term Knowledge Base".

In the following the expert system will be described in detail. If readers are interested in the "Prediction Engine", they are encouraged to read[18].

3.2.1. Expert system

The expert system is the main component of the driving system. It is responsible for deciding whether it is relevant to show the driver a driving recommendation or not. The decision is done on the basis of the information stored in the different "Knowledge Bases" and the predicted data. The decision task is split in two modules: The "Rule- and Data Element Selector" and the "Inference Engine". The "Rule- and Data Element Selector" module has the task to detect broken driving rules, changes of the current driving behaviour from the typical driving behaviour and uncommon driver conditions. On recognition of any of these irregularities, the irregularity will be passed with the corresponding data to the "Inference Engine" using the "Broken Rules Queue". The "Broken Rules Queue" contains the irregularity, for example a broken rule for energy-efficiency, and the data that caused the irregularity. Finally, the "Inference Engine" decides if it is necessary to show a driving recommendation relating to the irregularity in the "Broken Rule Queue". The decision is based on the reaction of the driver to already given recommendations. In the following the "Rule- and Data Element Selector" and the "Inference Engine" will be explained in detail.

3.2.1.1. Rule- and data element selector

As mentioned before, the "Rule- and Data Element Selector" module is responsible for detecting any breaking of a driving rule, deviation from the typical driving behaviour and any condition of the driver, which can be prejudicial for the driving task, like stress, anger and so on. Thus, it compares the data from the "Working Memory" and the predicted data against the driving rules, the driving profile and the car facts, using a rule-matching algorithm. If the "Rule- and Data Element Selector" recognises any broken rules, deviations from the typical driving behaviour or uncommon driver conditions, it puts the recognised irregularity with the corresponding information in to the "Broken Rule Queue" for further processing. The used rule-matching algorithm will be described in detail in Chapter 4.

3.2.1.2. Inference engine

The "Inference Engine" is responsible for deciding if it is necessary to generate and show each recommendation to the driver, with the goal of increasing the acceptance of the system by avoiding to bother the driver under certain circumstances. Therefore, it first takes an irregularity from the "Broken Rules Queue" and checks the corresponding information against the driver profile, especially the already given recommendations and the past reactions to them. The reactions to the already given recommendations are analysed during the next cycles by checking the changes of the values that are relevant for a specific recommendation. However, there are delays until the driver is able to notice and to react to a specific recommendation. In [19] the steering reaction time of drivers to a stimulus change, like opening of the door of a car parked on the roadside, has been examined. On the basis of the results, Summala recommend to reserve at least 3 seconds for drivers to respond by steering. However, the maximum response latency of a driver in [19] was about 4 seconds. Furthermore, the brake reaction time is examined in [20] for situations, in which the drivers have to brake suddenly and completely unexpectedly. The result showed an average reaction time of 0.9 seconds. However, 25% of the drivers had a longer reaction time than 1.2 seconds. Since the drivers have to react to given recommendations and the maximum reaction time of a driver to a stimulus change on the road side is 4 seconds, the "Inference Engine" will wait 5 seconds until it starts to analyse the driver reaction to a recommendation, as the driving system wants to give the driver additional time to become aware of the given

recommendation. The brake reaction time for an unexpectedly occurred traffic situation can be neglected for the estimation of the waiting time, as the recommendations do not appear suddenly or unexpectedly and do not require a sudden reaction.

The following example illustrates the approach: after showing the recommendation "shift the gear to the next, to keep the engine speed down", the "Inference Engine" waits until it starts to analyse the reaction. During the next cycles the gear is monitored and checked if the driver has shifted the gear. If the "Inference Engine" recognises a higher gear it will assume that the recommendation has been adhered. However, if no higher gear is noticed by the "Inference Engine", it will wait a certain time until it shows the same recommendation again, as it does not want to bother the driver. In case of repeated ignorance of that recommendation, the "Inference Engine" will decrease the generation frequency of that recommendation. This leads to an adaptation of the driving system to the individual driving behaviour whereby the driver needs are considered. Thus, the acceptance of the driving system can be increased, as recommendations, which are not necessary in the sense of the driver, can be avoided.

4. The improved rule match algorithm

The "Rule- and Data Element Selector", explained in Chapter 3.2.1.1, detects the breaking of driving rules or deviations from the normal driving behaviour or an uncommon driver condition using rule-matching algorithms. There are several rule-matching algorithms such as Rete, Treat and the Leaps algorithm. The Rete algorithm[21] compares rules against a certain data set. In the case of the presented driving system, rules are defined for the detection of broken driving rules, deviations of the current driving behaviour and any uncommon driver condition. A rule for a rule-matching algorithm is described by conditions and consequences. For example the rule saving energy consists of two conditions "current rpm > 2500" and "current gear < 6", where the value "6" represents the highest gear of the car. The consequence of the rule in the example is to "shift the gear" to keep the engine speed down, when the driver is not driving with the highest gear. The Rete algorithm is using a tree-structured network, also called Rete network, to represent the rules, see Fig. 2 (1) for an example. The tree consists of alpha nodes and beta nodes, where each alpha node represents one condition of the rule, for example "current rpm > 2500" is one alpha node. The beta nodes are used to join the alpha nodes. Thus, a beta node represents the joined condition, which was separated in to the alpha nodes, for example the beta node of our example would represent the joined conditions of "current rpm > 2500" and "current gear < 6". The Rete network stores a copy of the data within the nodes, where the data matched the condition of the node. Thus, every time when the data set changes, the data within the Rete network has to be updated. The update consists of two operations: deleting the old data and adding the new data. If all conditions of a rule are satisfied, the rule is passed to the conflict set, which consist all rules, whose conditions are

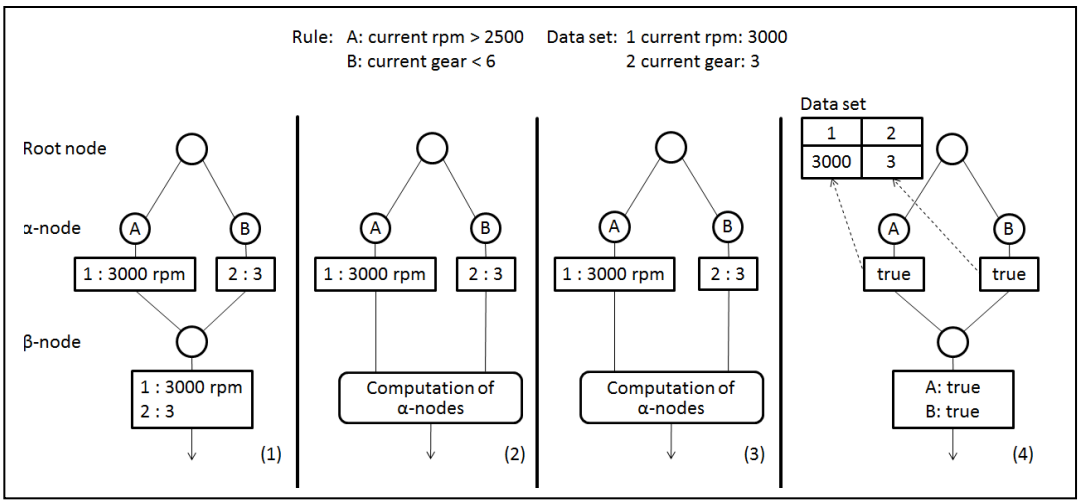

Fig. 2. The Rete network using the (1) Rete, (2) Treat, (3) Leaps and (4) the improved Rete approach

satisfied. According to the conflict set resolution strategy, for example rule ordering (first come, first served), the consequence of a rule is fired. The consequence of the rule used in the example before would be "shift the gear".

The Treat algorithm[22] is nearly identical to the Rete algorithm. However, the major advantage of the Treat algorithm is that it does not use beta nodes to join the alpha nodes, see Fig. 2 (2). Thus, the memory usage of Treat is lower. Instead, the relations of the alpha nodes to each other are recomputed when required. Thus, in the example in Fig. 2 (2) the alpha nodes "A" and "B" are recomputed, when the incoming data satisfies the nodes. The results of the computation include the rules, whose conditions are satisfied. These rules are also stored in the conflict set and are waiting for the activation of their consequence using a conflict set resolution strategy. According to Miranker[22] the Treat algorithm is more effective than the Rete algorithm, as it requires fewer comparisons to perform a binding of the data to the corresponding nodes. Moreover, during the deletion of data the Treat algorithm updates the alpha nodes and the conflict set directly, instead of recomputing alpha and beta nodes. Nayak et al.[23] evaluated the Treat algorithm, using four Soar† programs. The results of the evaluation showed that the Rete algorithm outperforms the Treat algorithm in most cases, especially when it is used in static structures, as Rete saves intermediate relations instead of recomputing them. According to Nayak et al. the evaluation results differ from Mirankers results in [22] because of the metrics used by Miranker. In the evaluation of the Treat algorithm, Miranker counted the number of comparisons, which may not "reflect the intrinsic differences between the match algorithms"[23].

The Leaps algorithm[24] is a rule match algorithm, based on Treat, see Fig. 2 (3). It is also using alpha nodes for representing the rule conditions and is also recomputing their relations when required. Moreover, the rules whose conditions are satisfied are stored in the conflict set, as well. The extension of the Leaps algorithm is the lazy evaluation of the conflict set, where only one rule is computed in each cycle instead of computing all possible rules. This allows increasing the rule firing rates and therefore decreasing the execution time, this is why it is suited for the usage at large databases.

The Rete algorithm is the basis algorithm for the Treat and Leaps algorithms. The Treat algorithm tries to improve the Rete algorithm by omitting the beta nodes and recomputing the relations of the nodes when required. The Leaps algorithm is the improvement of Treat by using a lazy evaluation to solve the conflict set. In the presented driving system a queue is used for collecting and ordering the broken rules, according to the "first in, first out" principle, as the driving system shows only one recommendation at the same time. Thus, a conflict set including a conflict set resolution strategy has not to be considered, why the Leaps algorithm can be neglected, as it only differs from the Treat algorithm regarding the resolution of the conflict set. Furthermore, as shown in [23], the Rete algorithm outperforms Treat in most cases, therefore the Rete algorithm will be the basis for the rule match algorithm used in the "Rule- and Data Element Selector" for detecting the broken driving rules, deviations from the typical driving behaviour and any uncommon driver condition.

The data set used in the "Rule- and Data Element Selector" consists of all information needed for the rule match algorithm. It stores the information of the current driving situation in tuples, which is defined by the name of a certain information and the corresponding value, for example "current rpm: 3000" and "current gear: 3". The amount of the tuples stored in the data set is not changing during a journey, as all relevant information is gained from the car, the user and the environment from the beginning of the journey. The tuples, respectively the values within the tuples, are updated by the "Data Layer" in the frequency of 100 Hertz, which means that the usage of the original Rete algorithm in the presented driving system would not be very efficient as it would have to delete the old and add the new value, in every Rete network with the frequency of 100 Hertz. To solve this issue, the Rete network has been modified, so that the alpha nodes are getting a pointer to the

† Soar, also cognitive architecture, is an architecture for systems capable of general intelligence

corresponding tuples during the initialisation of the Rete network. Thus, after the update of the data set the "Rule- and Data Element Selector" initiates the checking of the alpha node conditions. If a condition of the alpha node is satisfied, the result of the checking is stored in the alpha node and is passed to the corresponding beta node. The alpha and beta nodes store the results of the comparison of the alpha node conditions, instead of the redundant data stored in the alpha nodes. Fig. 2 (4) shows the Rete network of the improved Rete algorithm. This approach allows the faster processing of the Rete algorithm as it does not have to update the data of every node after every change of the data set, which is inefficient for the driving system.

The rules used to initialise the Rete network are defined in a text file with the file extension "DRR", which is the abbreviation for "driving rule". The "DRR"-file contains 13 energy-efficient and safety relevant driving rules, whose adherence are monitored by expert system. The energy-efficient driving rule "To shift as soon as possible" is described by shifting the gear at the latest by 2500rpm and when the current gear is not the highest gear. On the basis of these facts, the rule used for the Rete network has two conditions: "current rpm > 2500" and "current gear <= highest gear". Fig. 3 shows the description of the energy-efficient driving rule in the "DRR" file.

```
#rule
    "ecoRPM"
#when
    currentRpm > 2500 & currentGear < 6
#end
```

Fig. 3. An energy-efficient driving rule described in the "DRR" file

The begin of a rule description within the "DRR" file is indicated by key word "#rule", which is followed by the name of the described rule, "ecoRPM" in Fig. 3. The conditions of a rule are defined between the key words "#when" and "#end". If more than one condition is required to define a rule, the conditions are chained together with the symbol "&", as the Rete algorithm generate one alpha node for each condition. However, if a driving rule contains the logical operator "OR", the driving rule has to be modified so that it can be interpreted without using a logical "OR". So far, the Rete algorithm checks the presence of a condition. However, if it is needed to check the absence of a condition, it is possible to use negated conditions. A negated condition is indicated by using a "!" in front of the condition, for example "!(currentRpm > 2500)". Furthermore, the improved Rete algorithm allows defining a rule using a term, as the driving system has to calculate for example the minimum distance to the preceding car. The distance is calculated on the basis of the thumb rule "minimum distance = current car speed / 2". Thus, condition of the driving rule is defined with the following syntax: "current distance <= (current speed / 2)".

The Rete algorithm has been adapted and optimised for the usage in systems, whose data is changing frequently. It allows defining the driving rules within the "DRR" files by using logical expressions and terms. Furthermore, it has been optimised to do the rule matching in a more efficient way. The following chapter describes the evaluation of the improved Rete algorithm and their results.

5. Evaluation & results

The evaluation of the improved rule match algorithm was done by using a driving simulator. We captured the data of the virtual car during a journey of about 8 minutes on the driving simulator. The captured data consisted of information about the car like revolutions per minute, car speed, current gear and so on. On the basis of the captured data, we evaluated the improved and the original Rete algorithm using three driving rules. To measure the efficiency of the algorithms, we compared the accesses to the alpha and beta node memories and the comparisons of the changed data within the data set against the alpha and beta node conditions. During

the first measurement, the Rete network of the algorithms were initialised using the energy-efficient driving rule “shift as soon as possible at the latest by 2500rpm” Fig. 4 (1), which was used in Chapter 4 as an example. Thus, the Rete network of the improved and the original algorithm had the same structure as in Fig. 2 (4) and Fig. 2 (1). Furthermore, they consisted of two alpha nodes with the conditions “current rpm > 2500” and “current gear < 6” and one beta node, which joined the two conditions. In the following measurements of the evaluation, the safety relevant driving rules “do not exceed the speed limit” Fig. 4 (2) and “keep enough distance to preceding car” Fig. 4 (3) were used to initialise the Rete algorithms.

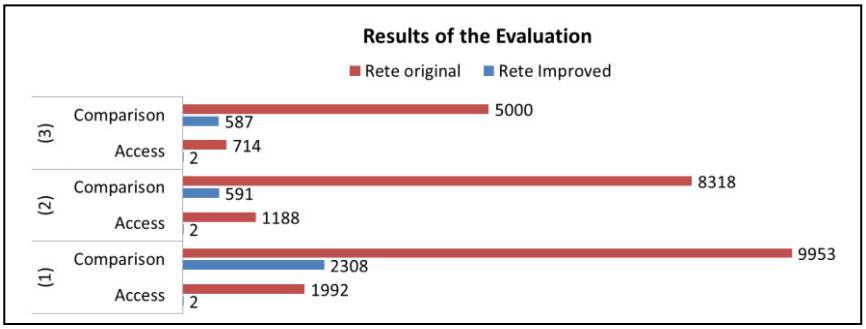

Fig. 4. The result of the evaluation with the rules: (1) CurrentRPM < 2500 & CurrentGear < 6, (2) CurrentSpeed > CurrentSpeedlimit and (3) CurrentDistanceToCar < CurrentSpeed/2

As shown in the results of the evaluation in Fig. 4, the improved rule match algorithm outperforms the original Rete algorithm, as it needed fewer comparisons of the node conditions and fewer accesses to the node memory. During the 8 min journey, the improved rule match algorithm needed only two accesses to the node memory to keep the nodes up to date, as it passes a pointer to the nodes, which point to the corresponding data of the variables “current rpm” and “current gear”. However, the original Rete algorithm needed more accesses to the alpha and beta memory, in which it removed the old and added the new data, which satisfied the condition of the node. Furthermore, the improved rule match algorithm needed fewer comparisons of the data against the different driving rule node conditions. In contrast, the original Rete algorithm needed about 14 times more to compare the data against the node conditions of the driving rule “do not exceed the speed limit”. In summary, the original Rete algorithm needed more comparisons of the data and more accesses to nodes. The difference of the performance is the effect of signalling the Rete network after every update of the data set to check the conditions of the nodes, instead of passing all changed data of the data set to the Rete network. The approach of the improved rule match algorithm saves comparisons, as the nodes get only pointers to the data, which are relevant for their comparison. The original Rete algorithm, however, passes all changed data within the data set to the Rete network, which then checks if the incoming data is relevant for a node and if the conditions of the nodes are satisfied by the incoming data. Thus, the usage of the original Rete algorithm is inefficient for the driving system presented in this paper, as its data set is changing frequently, which would lead to a frequent recomputation of the alpha and beta nodes.

6. Conclusion & further work

In this paper an improved rule match algorithm was introduced, which is used in the expert system of an energy-efficiency and safety relevant driving system. The goal of the driving system is to improve the driving behaviour in terms of energy-efficiency and safety by giving recommendations to the driver. Therefore, the improved rule match algorithm checks the incoming information about the car, driver and the environment against the typical driving behaviour, energy-efficient and safety relevant driving rules as well as the driver condition. This allows the driving system to create individually generated recommendations, which help

eliminating bad driving habits, while considering the driver needs. The rule match algorithm introduced in this paper is based on the Rete algorithm. However, the major difference between the two algorithms is passing a pointer to the relevant data instead of saving the data, which satisfied a condition of a node. This difference is the main reason why the performance of the Rete algorithm is poorer in the evaluated environment, as the matching of a frequently changing data set involves the frequent recomputation of the alpha and beta node memories in Rete, while the improved rule match algorithm is able to access the updated data using the pointer stored in the nodes.

Since the "Inference Engine" decides if it is necessary to show a recommendation to the driver, it has to be figured out, which algorithm fits the needs of the "Inference Engine". Furthermore, it has to be evaluated if the waiting time of 5 seconds after showing a recommendation to the driver satisfies the needs of the driver. Besides the evaluation of the algorithm, a user friendly concept has to be worked out, according to the usability guidelines for human-machine interfaces for in-vehicle systems[25]. Finally, the recommendations have to be displayed in a noticeable way without distracting the driver.

References

1. Yay M. *Elektromobilität*. Lang, Peter GmbH; 2010
2. German Statistical Office. *Verkehr – Verkehrsunfälle*. Wiesbaden. Germany; 2011
3. Fan X, Ji J, Zhang G. Impact of Driving Behavior on the traffic safety of Highway Intersection. *Proceedings of 3rd ICMTSA*; 2011
4. Haworth N, Symmons M. *Driving to Reduce Fuel Consumption and Improve Road Safety*. Australian Transport Council; 2001
5. Helms H, Lambrecht U, Hanusch J. Energieeffizienz im Verkehr. In *Energieeffizient*. Publisher Martin Pehnt. Springer; 2010
6. Van Mierlo J, Maggetto G, Van de Burgwal E, Gense R. Driving style and traffic measures – influence on vehicle emissions and fuel consumption. *Proceedings Institution of Mechanical Engineers*. Vol. 218 Part D: Automobile Engineering; 2004
7. Kia Website. *Eco Driving System*. http://kia-buzz.com/?p=252. Last visit: 07.02.2014
8. Lotan T, Toledo T. *An In-Vehicle Data Recorder for Evaluation of Driving Behavior and Safety*. Transportation Research Board of the National Academies; 2006
9. Bär T, Kohlhaas R, Zöllner J, Scholl K-U. Anticipatory Driving Assistance for Energy Efficient Driving. *ISTS*. Vienna; 2011
10. Corcoba Magana V, Munoz Organero M. Artemisa, Using an Android device as an Eco-Driving assistant. Cyber Journals: Multidisciplinary Journals in Science and Technology. *JMTC*. June Edition; 2011
11. Onken R. DAISY, an Adaptive, Knowledge-based Driver Monitoring and Warning System. *Vehicle Navigation & Information Systems Conference Proceedings*; 1994
12. Singh H, Bhatia JS, Kaur J. Eye Tracking based Driver Fatigue Monitoring and Warning System. *IICPE*. India; 2010
13. Sahayadhas A, Sundaraj K, Murugappen M. Detecting Driver Drowsiness Based on Sensors: A Review. *Sensors*. Vol.12 Issue 12. Basel. Switzerland; 2012
14. Lee CC. Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I. In *IEEE Transaction on SMC*. Vol. 20; 1990
15. Shin H, Jung S, Kim J, Chung W. Real Time Car Driver's Condition Monitoring System" *IEEE SENSORS 2010 Conf.*; 2010
16. Kumar M, Weippert M, Vilbrandt R, Kreuzfeld S, Stoll R. Fuzzy Evaluation of Heart Rate Signals for Mental Stress Assessment. *IEEE Transactions on Fuzzy Systems*. Vol. 15, No. 5; 2007
17. Brockwell PJ, Davis RA. *Introduction to Time Series and Forecasting*, Second Edition, Springer; 2002
18. Yay E, Martínez Madrid N. SEEDrive: An Adaptive and Rule Based Driving System. *Proceedings of 9th Internation Conference on Intelligent Environments*; 2013
19. Summala H. Driver/Vehicle Steering Response Latencies. *HFES*; 1981
20. Johansson G, Rumar K. Drivers' Brake Reaction Times. *HFES*; 1971
21. Forgy CL. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Expert Systems*, 1990
22. Miranker DP. TREAT: A Better Match Algorithm for AI Production Systems, *AAAI'87 Proceedings*; 1987
23. Nayak P, Gupta A, Rosenbloom P. Comparison of the Rete and Treat Production Matchers for Soar (A Summary). *AAAI*; 1988, p 693-698
24. Miranker DP, Brant DA, Lofasso B, Gadbois D. On the Performance of Lazy Matching in Production Systems. *AAAI-90 Proceedings*; 1990
25. European Commission. Commission Recommendation on safe and efficient in-vehicle information and communication systems: update of the European Statement of Principles on human machine interface. *Official Journal of the European Union*; 2007