2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Applying Neural-Symbolic Cognitive Agents in Intelligent Transport Systems to reduce CO_2 emissions

Leo de Penning, Artur S. d'Avila Garcez, Luis C. Lamb, Arjan Stuiver, and John-Jules Ch. Meyer

***Abstract*—Providing personalized feedback in Intelligent Transport Systems is a powerful tool for instigating a change in driving behaviour and the reduction of CO_2 emissions. This requires a system that is capable of detecting driver characteristics from real-time vehicle data. In this paper, we apply the architecture and theory of a Neural-Symbolic Cognitive Agent (NSCA) to effectively learn and reason about observed driving behaviour and related driver characteristics. The NSCA architecture combines neural learning and reasoning with symbolic temporal knowledge representation and is capable of encoding background knowledge, learning new hypotheses from observed data, and inferring new beliefs based on these hypotheses. Furthermore, it deals with uncertainty and errors in the data using a Bayesian inference model, and it scales well to hundreds of thousands of data samples as in the application reported in this paper. We have applied the NSCA in an Intelligent Transport System to reduce CO_2 emissions as part of an European Union project, called EcoDriver. Results reported in this paper show that the NSCA outperforms the state-of-the-art in this application area, and is applicable to very large data.**

***Index Terms*—Neural-Symbolic Learning and Reasoning, Driver modelling, Deep Learning, Restricted Boltzmann Machines (RBM),**

I. Introduction

RESEARCH has shown that providing feedback is a powerful tool for instigating a behaviour change (e.g. [1], [2], [3]). As part of an EU project on the reduction of CO_2 emissions, called EcoDriver, we focused on innovations in feedback advice strategies and Human-Machine Interface (HMI) solutions to maximize system effectiveness and acceptance of an Intelligent Transport System that supports drivers to reduce their CO_2 emissions. For example, we applied an adaptive feedback strategy to tailor the feedback and HMI, based on the driving style of the driver (see Figure 1). This requires an automated driver type detection system that is able to reason online about observed driving behaviour and classify this behaviour in terms of several characteristics of the driver (e.g. learning vs. performance oriented, social vs. self oriented) using large amounts of real-time and noisy vehicle data (e.g. steering angle, speed, rpm, gear, brake). Although the use of semi-controlled environments, like an instrumented car, simplifies the data and knowledge acquisition, the building of a model or intelligent autonomous agent that is capable of dealing with the many complex temporal relations in the observed data is a very difficult task, in particular in dynamic and non-stationary environments [4], [5].

Figure 1. EcoDriver human-machine interface for personalized feedback on CO_2 emissions.

In this paper, we address the problems by applying the architecture and theory of the Neural-Symbolic Cognitive Agent (NSCA) described in [6], [7] to develop a robust model for driver type detection from real-time vehicle data. We will show the NSCA is capable of deep learning and reasoning about complex dynamic temporal relations, but also represent an agent's knowledge in symbolic form for explanation and validation purposes, describing its decisions and providing feedback to the user. This is achieved by taking advantage of neural-symbolic integration [8], using the networks for performing robust learning and adaptation, and symbolic knowledge extraction for representing the temporal relations explicitly and for qualitative reasoning. We also provide additional proof showing the NSCA is capable of modelling any temporal logic program. The result is an agent model that is capable of efficient online learning and reasoning in complex, dynamic and non-stationary environments. We discuss the effective use and results of the NSCA as part of the driver type detection system, and refer to several other real-world applications of the NSCA (e.g. automated assessment

Leo de Penning is with the Department of Earth, Life and Social Sciences, TNO, Soesterberg, The Netherlands, e-mail: leo.depenning@tno.nl.
Artur S. d'Avila Garcez is with the Department of Computer Science, City University London, UK, e-mail: a.garcez@city.ac.uk.
Luis C. Lamb is with the Instituto de Informatica, UFRGS, Porto Alegre, RS, Brazil, e-mail: LuisLamb@acm.org.
Arjan Stuiver is with the Department of Earth, Life and Social Sciences, TNO, Soesterberg, The Netherlands, e-mail: arjan.stuiver@tno.nl
John-Jules Ch. Meyer is with the Department of Information and Computing Sciences, Universiteit Utrecht, Netherlands, e-mail: J.J.C.Meyer@uu.nl.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

in driver training, behaviour recognition in video), to show that NSCA can be useful at performing online learning and explanation in different real-world domains, indicating the reach of the approach.

II. Preliminaries

The construction of effective cognitive agent models is a long standing research endeavour in artificial intelligence, cognitive science, and multi-agent systems [4], [9]. One of the main challenges toward achieving such models is the provision of integrated cognitive abilities, such as learning, reasoning and knowledge representation. Recently, cognitive computational models based on artificial neural networks have integrated inductive learning and deductive reasoning, see e.g. [8], [10]. In such models, neural networks are used to learn and reason about (an agent's) dynamic knowledge about the world, represented by symbolic logic. In order to do so, algorithms map logical theories (or knowledge about the world) T into a neural network N which computes the logical consequences of T. This provides also a learning system in the network that can be trained by examples using T as background knowledge. In agents endowed with neural computation, induction is typically seen as the process of changing the weights of a network in ways that reflect the statistical properties of a data set, allowing for generalizations over unseen examples. In the same setting, deduction is the neural computation of output values as a response to input values (stimuli from the environment) given a particular set of weights. Such network computations have been shown equivalent to a range of temporal logic formalisms [6]. The NSCA architecture described in this paper is based on this approach and uses temporal logic as theory T and a Restricted Boltzmann Machine (RBM) as neural network N [11].

An RBM represents a stochastic neural network with visible units **v**, that represent input variables (or hidden-unit activations of lower-layer RBMs), and hidden units **h**, that represent the likelihood of certain activation patterns in **v**. There are symmetric weighted connections between the hidden and visible units with weights W, but no connections within the hidden units or visible units. The weights can be trained to model a joint probability distribution over **h** and **v** (see Equation 1, where **b** and **c** denote the biases of the hidden and visible units and $\sigma(x)$ the logistic sigmoid function). This particular configuration makes it easy to compute the conditional probability distributions, when **v** or **h** is fixed (Equation 2), enabling the reconstruction of input data **v'** based on partial information in **v**. This is done by sampling the conditional probability distribution in Equation 2, where $h'_j = 1$ with $p(h_j|\mathbf{v})$ (and $h'_j = 0$ otherwise), and calculating the reconstructed data **v'**, where $v'_i = P(v_i|\mathbf{h'})$.

$$-\log P(\mathbf{v},\mathbf{h}) \propto E(\mathbf{v},\mathbf{h}) = -\mathbf{c}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T W\mathbf{v} \quad (1)$$

$$P(h_j|\mathbf{v}) = \sigma(b_j + w_j^T\mathbf{v}) \quad (2)$$

The RBM can be trained using Contrastive Divergence learning [12]. This learning algorithm tries to minimize the difference between **v** and $\mathbf{v}'$ by changing the weights using a Hebbian-like learning rule such that $\triangle W \cong \mathbf{v}\cdot\mathbf{h} - \mathbf{v}'\cdot\mathbf{h}'$, with the network in the long run learning to approximate the joint probability distribution $P(\mathbf{v},\mathbf{h})$.

III. Neural-Symbolic Cognitive Agent

In its core, the NSCA uses a Recurrent Temporal Restricted Boltzmann Machine (RTRBM) for online learning and reasoning (see Figure 2). An RTRBM is an RBM where each hidden unit has recurrent connections to the hidden unit activations at time $t-1$ [13]. This enables the NSCA to model hypotheses H, that are represented by the hidden units, on temporal relations between beliefs B, that are represented by visible units, by setting the weights of the RTRBM [7]. A major benefit of the NSCA is that these hypotheses can both be modelled by hand as prior knowledge, similar to other logic-based reasoning systems (e.g. see [14]), as well as learned from observation, allowing the refinement of prior knowledge. For example, the NSCA can model a rule r like 'When it rains, the grass will get wet' by defining a hypothesis H_1, represented by a hidden unit, that implies a temporal relation between the beliefs B_1 (i.e. 'it rains') and B_2 (i.e. 'grass is wet'), represented by visible units. Since B_1 at time t will result in B_2 at a later time $t+n$ (i.e. the grass will get wet and remain wet for a while after it started raining), another hypothesis H_2 is used to denote that it is currently raining. Both H_1 and H_2 are represented by hidden units with recurrent connections in the RTRBM and are used to establish the desired temporal link between B_1 and B_2. As described in section IV, this can be expressed by two temporal logic rules: $H_1 \leftrightarrow B_2 \wedge \bullet H_2$ and $H_2 \leftrightarrow B_1 \wedge \bullet H_2$ (where $\bullet$ means 'at time $t-1$'; everything else happens at time t).

When the NSCA then observes B_1 (i.e. activating the related visible unit with some probability or real value), it can calculate the probability that it is raining (i.e. the activation of H_2) by forward propagation of the belief activation to the hidden unit representing H_2. Similarly, this can be done for H_1 at time $t+1$ using the probability of H_2. We refer to this as **deduction** in the NSCA. Deduction (Figure 2, right) is similar to Bayesian inference, where for all hypotheses H, the probability is calculated that the hypotheses are true given the observed beliefs b and the previously applied hypotheses H^{t-1} (i.e. $P(H|B=b,H^{t-1})$). From the posterior probability distribution, the RTRBM selects the most likely hypotheses h' using random sampling, denoted as $h' \sim P(H|B=b,H^{t-1})$, whereby at each time, a random number n_i is generated for each hypothesis H_i in the interval [0,1]. If $h'(i) > n_i$, where $h'(i)$ is the activation value of hidden unit h_i, then hypothesis H_i is selected (i.e. the activation value of h_i is set to 1); otherwise, h_i is set to zero.

So based on the probability of H_1, NSCA can assume this hypothesis is actually true and applicable to the current observation by setting the activation of the related hidden unit to 1.0. Then, using backward propagation to the visible units and recurrent connections, NSCA can calculate the probabilities of all beliefs and previous activations of the hidden units, given the assumptions made. As it already observed that it is raining (H_2), it will infer that the grass is wet (B_2). We refer to

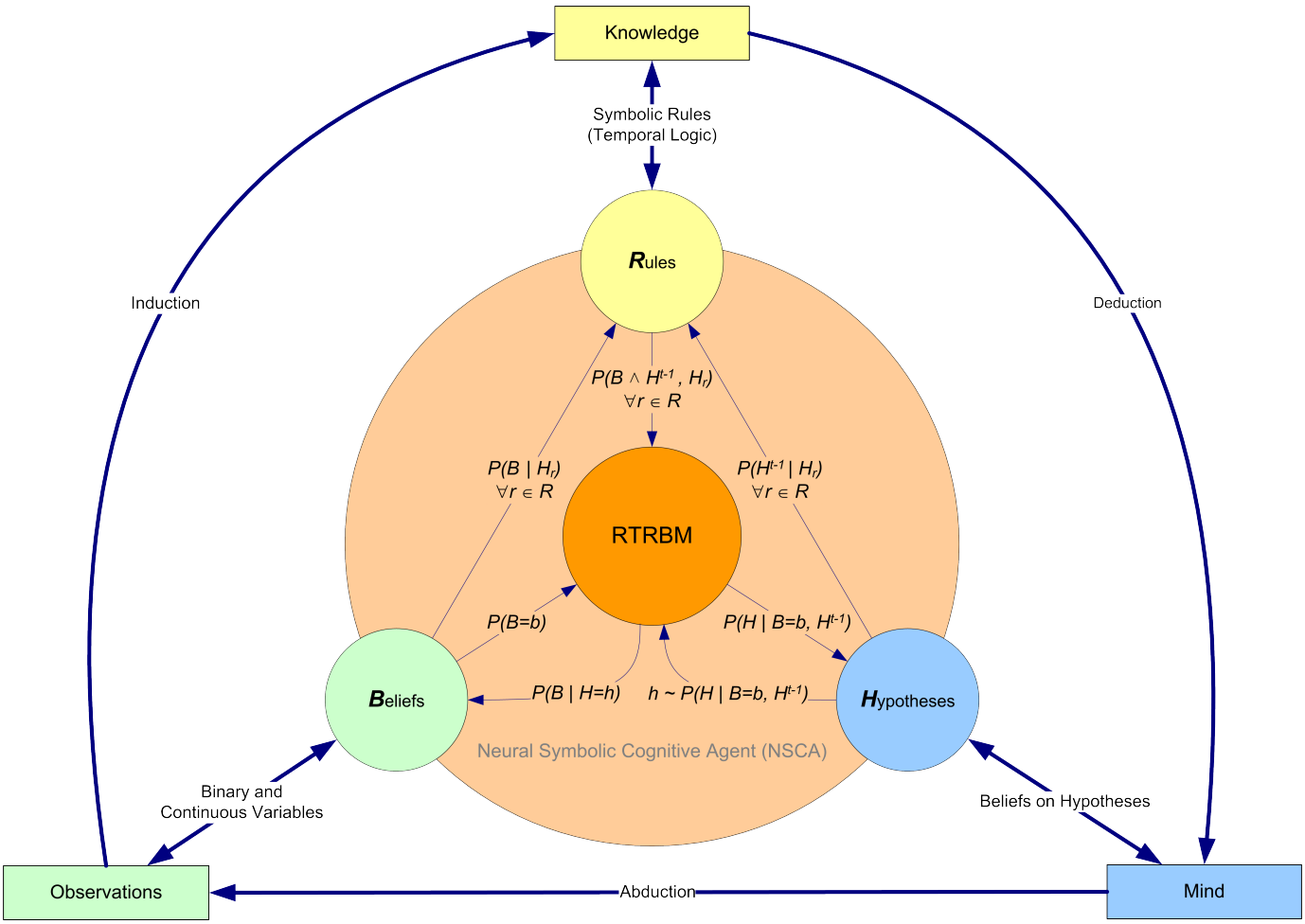

Figure 2. Neural-Symbolic Cognitive Agent architecture that combines robust learning from observation and probabilistic temporal reasoning using an RTRBM and enables symbolic knowledge encoding and extraction based on its Bayesian inference model. A complete cycle (e.g. deduction, abduction and induction) is performed online (in near real-time) and includes updating the RTRBM based on the difference between the observed beliefs $P(B = b)$ and abduced beliefs $P(B|H)$. Adaptation in the RTRBM is controlled by a learning parameter that controls the training and testing ratios in the NSCA.

this process as **abduction** in NSCA. Via abduction (roughly, the process of finding a posteriori the best explanations for an observation; see Figure 2, bottom), NSCA can infer the most likely beliefs B based on hypothesis H by calculating their conditional probability $\mathbf{b}$ (or real value in case of real-valued beliefs) from the RTRBM, i.e. $\mathbf{b} = P(B|H)$. The differences between the observed and inferred beliefs are then used by NSCA to determine the implications of the observed beliefs and applied hypotheses (e.g. an assessment of high-order driving skills based on observed driving behaviour or detection of human behaviour in video).

Finally, suppose NSCA observes that the grass is wet (B_2) every time it observed H_2, and these observations often triggered H_1 through forward propagation. The weights between the related visible and hidden units will be updated so that these concepts will become strongly correlated, possibly forming a new relation. We refer to this as **induction** in NSCA. Learning of new relations through induction can be achieved by using the differences between the observed and inferred beliefs to strengthen the correlation between the selected hypotheses H and the observed beliefs B (Figure 2, left). NSCA does so by updating the weights in the RTRBM using both Contrastive Divergence and Backpropagation-Through-Time as done in [13].

The NSCA architecture also enables the construction of multiple NSCAs in a kind of multi-agent system that is capable of deep learning and reasoning, where each NSCA can learn and represent higher-order temporal relations on hypotheses of other NSCAs by observing the probabilities on these hypotheses (depicted as the current state of 'mind' in Figure 2). Such a layered approach results in a Deep Belief Network (or Deep Boltzmann Machine when RBMs are used), and is capable of meta-level learning and reasoning [15]. Also the rules extracted from higher-order NSCAs represent meta-level rules on lower-order hypotheses (e.g. $H_1^{meta} \leftrightarrow H_1 \wedge \neg H_3$ denotes that there is a disjunctive relation between occurrences that the the grass is wet, because it has rained, or that it is wet, because the sprinkler was on).

So far, we have explained the bottom part of the model in Figure 2, showing how beliefs and hypotheses are modelled in the RTRBM. In the next section, we will explain the top part of the model, showing how symbolic rules can be encoded and extracted from the RTRBM.

IV. Temporal Knowledge Representation

Temporal knowledge can be encoded in the RTRBM in the form of temporal logic clauses that describe equivalences between hypotheses and beliefs over time. Therefore, they are adequate for representing non-stationary and dynamic knowledge, as will be exemplified later. In NSCA, $H_1 \leftrightarrow B_1 \wedge B_3 \wedge \bullet H_1$ is used to denote that hypothesis H_1 holds at time t if, and only if, beliefs B_1 and B_3 hold at time t and hypothesis H_1 holds at time $t-1$, where we use the previous time temporal logic operator $\bullet$ to denote $t-1$.

We consider a broad set of past and future temporal logic operators. The past operators include the usual representation of previous time (denoted by $\bullet$), always in the past ($\blacksquare$), sometimes in the past ($\blacklozenge$), and since (S). The dual operators in the future are, respectively, the next time operator (denoted by $\circ$), always in the future ($\Box$), sometimes in the future ($\Diamond$) and until (U). A logical expression α is called a temporal formula if and only if one of the following is true: (i) α is a propositional variable, (ii) $\alpha = \bullet\beta$, $\alpha = \blacksquare\beta$, $\alpha = \blacklozenge\beta$ or $\alpha = \beta S \delta$, where β and δ are temporal formulas, (iii) $\alpha = \circ\beta$, $\alpha = \Box\beta$, $\alpha = \Diamond\beta$ or $\alpha = \beta U \delta$, where β and δ are also temporal formulas. A *literal* λ is either a temporal formula (α) or the negation of a temporal formula ($\neg\alpha$). A *temporal clause* is an expression of the form $\lambda_1 \wedge \lambda_2 \wedge ... \wedge \lambda_n \rightarrow \alpha$, where α is a temporal formula and $\lambda_i (1 \leq i \leq n)$ are literals. A temporal logic program R is a set of temporal clauses.

A fixed-point semantics for the temporal programs R as defined above was provided in [6], following the natural intuition of the temporal operators. Each formula containing any of the above operators can be translated into an equivalent formula containing only the previous-time operator. The reader is referred to [16], [6] for the details and translations of the operators into formulas containing the previous-time operator only. As an example, the proposition $\alpha S \beta$ denotes that a proposition α has been true since the occurrence of proposition β. This can be translated into the following propositions: $\beta \rightarrow \alpha S \beta$ and $\alpha \wedge \bullet(\alpha S \beta) \rightarrow \alpha S \beta$, where $\alpha S \beta$ is a literal. In the RTRBM, α and β are modelled by visible units, $\alpha S \beta$ by a hidden unit and $\bullet(\alpha S \beta)$ is modelled by a recurrent connection to the hidden unit activation at time $t-1$. Given network N and program R, we say that N computes R if N approximates the fixed-point semantics for R upon the application of the rule encoding algorithm described below.

A general method for encoding and extracting symbolic rules from symmetric connectionist networks has been proposed in [17]. This method maps logical rules to the energy function of such networks by adding a penalty to the rules, as follows. An energy function $E(x)$ can be shown equivalent to sets of pairs of logic formulas and real numbers $< p_i, f_i >$. The real number p_i is called a "penalty" and the set of pairs of formulas and penalties is known as a penalty logic well-formed formula or PLOFF. A connectionist system and a rule set are equivalent if and only if there exists a ranking function V_{rank} for the latter such that $V_{rank}(x) = E(x) + c$, where c is a constant. In the case of an RBM with a visible unit v_i and a hidden unit h_j, a corresponding PLOFF would be $(\langle w_{ij}, v_i \wedge h_j | w_{ij} > 0 \rangle, \langle -w_{ij}, \neg(v_i \wedge h_j) | w_{ij} < 0 \rangle)$, where w_{ij} is the value of the weight between v_i and h_j, with as ranking function the sum of the penalties of the pairs whose formulas are violated given truth-values for $\mathbf{h}$ and $\mathbf{v}$. As can be seen, the penalty p_i is directly related to the weight of the connections between the beliefs and hypotheses.

NSCA extends the Penalty Logic approach to the case of RTRBMs in order to encode and extract rules from these networks. The penalties can be seen dually as the strength or confidence value of a rule. For clarity, we use the term confidence and normalize this value so that each rule will have a confidence between 0 and 1. Because NSCA uses a Restricted Boltzmann Machine, in which hidden units are conditionally independent of each other, we can optimize the rule insertion and extraction algorithms to make them simpler and much more efficient. Due to this conditional independence, we can treat each hidden unit as a separate rule (i.e. as an expert on some feature of the belief space). This means that our extraction and encoding can be applied to each hidden unit separately, as follows.

A. Extraction Algorithm

NSCA can extract temporal knowledge acquired through inductive learning, from a trained RTRBM in the form of symbolic rules R obtained directly from the network's weights W. To extract a temporal logic program Ψ from an RTRBM N we need to find the states of N that lower the total energy in its energy function. Such extraction can help explain the computation and provide feedback to the user about the learning process, as exemplified later. This means finding the states in the network N that maximize the likelihood of each clause r encoded in N. Once N is stable we can extract these clauses one at a time by clamping the maximum probability of hypothesis H_r to the related hidden unit h_r in N, and inferring the associated beliefs B and previous-time propositions H^{t-1} from the RTRBM using random sampling of the conditional probability distribution, i.e. $b_r = P(B|H_r)$ and $h_r^{t-1} = P(H^{t-1}|H_r)$, where $\mathbf{b_r}$ is a vector containing the probabilities of each belief associated with H_r, and $\mathbf{h_r^{t-1}}$ is a vector containing the probabilities of each previous-time hypotheses associated with H_r.

If we repeat the above process for each hidden unit, we can construct a temporal logic program Ψ using the following equations (where k is the number of beliefs, m the number of hypotheses and w_{ir} is the weight of the symmetric connection between the related visible unit v_i and hidden unit h_r, and w'_{lr} is the weight of the recurrent connection between the previous hidden unit activation h_l^{t-1} and hidden unit h_r in the RTRBM).

$$\Psi = \{\langle c_r : H_r \leftrightarrow \bigwedge_{i=1}^{k} \theta_r^{(i)} \bigwedge_{l=1}^{m} \rho_r^{(l)} \rangle, \forall r \in R\} \tag{3}$$

$$\theta_r^{(i)} = \begin{cases} B_i & \text{if } w_{ir} > 0 \wedge b_r(i) > 0.5 \\ \neg B_i & \text{if } w_{ir} < 0 \wedge b_r(i) > 0.5 \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

$$\rho_r^{(l)} = \begin{cases} \bullet H_l & \text{if } w'_{lr} > 0 \wedge h_r^{t-1}(l) > 0.5 \\ \bullet \neg H_l & \text{if } w'_{lr} < 0 \wedge h_r^{t-1}(l) > 0.5 \\ \emptyset & \text{otherwise} \end{cases} \tag{5}$$

$$c_r = P(H_r|B = \mathbf{b_r}, H^{t-1} = \mathbf{h_r^{t-1}}) \quad (6)$$

The literals in $\theta_r^{(i)}$ associated with the beliefs in clause r, are calculated using Equation 4 and depend on the weight w_{ir}. A negative weight w_{ir} makes the probability of H_r increase when the probability of B_i decreases and thus the probability of $\neg B_i$ increases. The inverse applies to a positive weight. When w_{ir} is 0 or $b(i) < 0.5$, belief B_i has no significant influence on the hypothesis and can be left out. A similar approach is used to extract the literals in $\rho_r^{(l)}$ for H_r^{t-1}. Finally, Eq. 6 shows how we calculate the confidence value c_r of rule r, denoting the strength or "penalty" of the equivalence relation, as done in [17]. This confidence value is based on the notion of Bayesian credibility described in [18] and is calculated in a similar way.

In case of beliefs with continuous values, literals in $\theta_r^{(i)}$ can be extracted using the same approach in the form of $B_i < b_r(i)$ when $w_{ir} < 0$ and $B_i > b_r(i)$ when $w_{ir} > 0$. This is general enough to account for the use of linguistic variables, like $weather = good$, where a threshold value is used to define if B_i should be true or false, and when a variable is naturally associated with a range in the real numbers, like $temperature = high$ if $B_1 > 40°\mathrm{C}$. Notice that, for efficiency, intervals are represented by the combination of two beliefs, like $temperature = mild$ if $temperature_{min} > 15°\mathrm{C} \wedge temperature_{max} < 25°\mathrm{C}$.

B. Encoding Algorithm

The NSCA can also encode prior knowledge in the form of symbolic rules R into the weights of the RTRBM by translating the rules into a joint probability distribution on beliefs B, hypotheses H_r for each rule $r \in R$ and their probabilities at a previous time H^{t-1}. To encode a set of temporal clauses R containing the entire range of temporal operators above into an RTRBM, first we rewrite each temporal clause $r \in R$ into a rule that uses only the previous-time temporal operator $\bullet$, as done in [6]. Then, for each r, we assume that its completion holds and write it as an equivalence relation between a so-called hypothesis for that rule, H_r, and the literals that denote beliefs B and previous-time propositions $\bullet H$. This creates a rule in the form shown in Equation 3, where θ's are beliefs and ρ's are previous-time propositions, to which a confidence value c_r can be attached (if available); otherwise, a confidence value of 1 is used.

To translate a set of temporal logic clauses R in the form of Ψ into an RTRBM, we use the *Contrastive Divergence* learning algorithm [13] and perform the following steps for each $r \in R$:

1) Add hidden units h_r, $h_{1..m}$ to the RTRBM (if they do not exist already) to represent hypothesis H_r and previous-time propositions $\bullet H_{1..m}$.
2) Add visible units $v_{1..k}$ to the RTRBM (if they do not exist already) to represent each belief literal $B_{1..k}$ in the clause.
3) For any positive literal B_i, set visible unit $v_i = 1$. For any negative literal $\neg B_i$ in the clause, set visible unit $v_i = 0$. Similarly, set previous hidden unit $h_l^{t-1} = 1$, for any $\bullet H_l$ in the clause, and $h_l^{t-1} = 0$, for any $\bullet \neg H_l$.
4) Randomize the weights connecting the visible units $\mathbf{v}$ and hidden units $\mathbf{h}$ to initialize the encoded clause with a random probability distribution.
5) Apply the contrastive divergence algorithm [13] to optimize the joint probability distribution $P(H_r = c_r, B = \mathbf{v}, H^{t-1} = \mathbf{h^{t-1}})$ and update the weights accordingly with a single shot update (i.e. learning rate = 1.0) assuming non-conflicting clauses. In case of conflicting or ambiguous clauses use a lower learning rate and iterate a couple of times over all clauses to find an optimum.

Theorem 1. *For any temporal logic program R there exists an RTRBM N such that N computes R.*

Proof: The translation of temporal formulas into clauses containing only the previous-time operator $\bullet$ is sound w.r.t. a fixed-point semantics for R [8], [16], [6]. For each rule r of the form shown in Eq. 3, assume that a first time point $t = 0$ exists without loss of generality. Given an assignment of truth-values to $H_{1..m}$ at time $t = 0$ and an assignment of truth-values to the beliefs $B_{1..k}$ in r, we have that H_r will hold with probability c_r at time $t = 1$. Inductive step: at time $t = n$, either N is stable in which case H_r has probability c_r of being activated, or a value for H_r is inferred from B and $\bullet H$. At time $t = n + 1$, from the encoding algorithm and network symmetry, $\bullet H$ will be inferred and H_r will be activated with arbitrary confidence level c_r. This holds for any H_r in the set of rules with an arbitrary chain from $\bullet H$ (through $\mathbf{h^{t-1}}$) to H_r. Hence, N computes R with arbitrary confidence $\sum c_r$. This completes the proof. ■

V. Driver Type Detection

The NSCA has already been applied for the modelling, recognition, description and explanation of human behaviour in various real-world applications. For example, the NSCA has been shown capable of assessing high-order driving skills (e.g. safe driving) by learning from observations of real-time dynamic simulation data (e.g. position and orientation of vehicles, gear, steering wheel angle, etc.) [7], and is capable of detecting useful human behaviour (e.g. chase, exchange, jump, etc.) observed in video streams given low-level visual features (e.g. bounding box properties of detected objects), and provide a meaningful temporal logic-based description of the behaviours in terms of these features [19].

In this paper we discuss the application of the NSCA in an intelligent transport system to reduce CO_2 emissions as part of a large scale EU project, called EcoDriver. Here the NSCA was used to classify the driving behaviour of a driver in terms of several driver type indicators (i.e. learning vs. performance oriented, social vs. self oriented, and sportive driving) based on the observation of real-time vehicle data from the CAN-bus of an instrumented car (e.g. speed, gear, indicators, steering angle, etc.). This information is used by the system to select among predefined feedback strategies in order to make drivers more aware on the use of fuel and CO_2 emissions based on

their driving style. Therefore we applied two NSCAs in a multi-agent configuration (see Figure 3). One agent to detect low-level driving behaviour (e.g. making a right turn, taking an exit, passing a car on the highway) from temporal relations in observed vehicle data (i.e. normalized between 0 and 1), and another agent to detect the driver type indicators from temporal relations in the low-level driving behaviour detected by the first agent (i.e. activation in the hidden units that reflect the conditional probabilities of specific driving behaviour based on observed vehicle data).

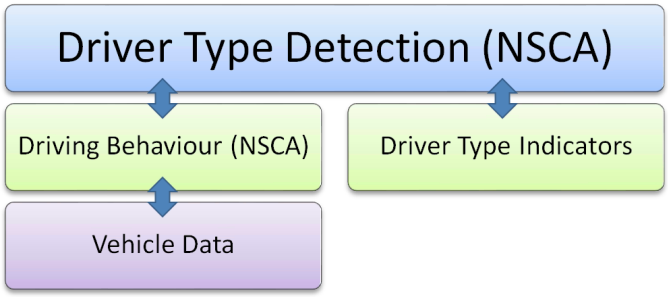

Figure 3. Multi-agent NSCA configuration for adaptive driver type detection.

Both models were trained sequentially (driving behaviour first) on vehicle data, collected using an instrumented car, and driver type indicators, determined from a questionnaire, for 29 different drivers (i.e. 70% male and 30% female between age 20 and 70 and driving experience between 5 and 50 years) driving a predefined route of 11km with urban, rural and highway parts. This resulted in 513,365 samples on 18 vehicle parameters and 3 related driver type indicators for 25 drivers (i.e. 4 drivers were left out because of incorrect data). After 2.5 hours of training the system reached a mean accuracy (i.e. F1-measure[1] measured per sample at 10Hz) for each participant and driver type of 0.528 (i.e. average precision and recall are resp. 0.720 and 0.733). Compared to other work on classification of driver behaviour, the module performs very well on the collected data. For example, [20] reports an average accuracy for driver identification of 0.295 using a Hidden-Markov Model (HMM) and [21] reports an average accuracy of 0.502 using K-means clustering. Figure 4 shows the F1-measure for each driver and driver type indicator separately. As can be seen the NSCA is able to detect driver types very accurately for some drivers (~ 0.9), showing the performance of applying multiple NSCAs in a multi-agent deep network configuration.

As can be seen in figure 4, the system had difficulty classifying certain drivers correctly. Especially the detection of social and sportive behaviour was difficult. Main reason for this is the limited dataset (i.e. only 25 drivers), where minor inconsistencies in the data (e.g. driver type indications from the questionnaire contradicted with the actual observed driving behaviour) can have a huge impact on the model performance. To improve the system and overcome these problems, we can use the online learning capability of the NSCA to further adapt the models based on new data gathered during operation (e.g. during field-tests or even commercial use). Another approach would be to let a group of experts on driving behaviour validate and modify the models directly, using the NSCA extraction and encoding algorithms (see section IV). The benefit of using an NSCA is that both approaches can complement each other in an iterating process, combining the robustness and flexibility of online learning with the expressive power of a symbolic knowledge representation.

To demonstrate this approach we have applied the extraction algorithm in section IV-A to extract symbolic rules on low-level driving behaviour from the Driver Behaviour NSCA (see Figure 3). Figure 5 shows a visual representation of these rules in the form of a matrix that represents the temporal relations between learned hypotheses and observed beliefs. This provides a more compact representation compared to a temporal logic program and thus suitable for models with many temporal relations. In Figure 5, the vertical axis depicts the beliefs (i.e. vehicle parameters and temporal relations $\bullet H$), and the horizontal axis depicts the hypotheses H, where red denotes a high positive correlation and blue denotes a high negative correlation. The bottom row depicts the confidence c of the hypotheses (i.e. Bayesian credibility) as calculated by Eq. 6. A similar representation can also be extracted from the Driver Type Detection NSCA, supporting the validation of the complete driver type detection system using human-readable symbolic rules. In this case the rules describe temporal relations between hypotheses on driving behaviour (as modelled by the Driving Behaviour model depicted in Figure 5) and the driver types. For sake of completeness we also provide two examples of rules on driving behaviour in temporal logic form (see Eqs. 7 and 8; we only provide the first few $\bullet H$ literals for the sake of conciseness):

$$0.99 : H_{26} \leftrightarrow RAWR > 0.81 \wedge RelativeVelocity > 0.58 \wedge aLat < 0.4 \wedge acceleration > 0.65 \wedge \bullet H_1 \wedge \bullet \neg H_2 \wedge \bullet \neg H_3 \wedge ... \quad (7)$$

$$1.00 : H_{27} \leftrightarrow RAWR > 0.85 \wedge RelativeVelocity < 0.57 \wedge aLat > 0.4 \wedge acceleration < 0.62 \wedge \bullet H_1 \wedge \bullet H_2 \wedge \bullet \neg H_3 \wedge ... \quad (8)$$

Based on the extracted knowledge, experts on driving behaviour may conclude for example, that these rules denote the difference between taking a right exit from the highway (i.e. Eq. 7), where the right indicator is on ($RAWR > 0.81$), there is a high relative velocity ($RelativeVelocity > 0.58$) and acceleration ($acceleration > 0.65$), but not so much lateral acceleration ($aLat < 0.4$), and taking a right turn in an urban area (i.e. Eq. 8), where the right indicator is on ($RAWR > 0.85$), but there is no high relative velocity ($RelativeVelocity < 0.57$) or acceleration ($acceleration > 0.65$), but more lateral acceleration ($aLat > 0.4$). To strengthen these rules they can add an additional constraint to Eq. 8 on the maximum speed in a right turn (e.g. $speed < 50$ in km/h) and then encode this rule back in the NSCA, thus adding expert knowledge to the driving behaviour model and hopefully improving its performance. Vice versa, the NSCA can also be used to fine-tune or even falsify existing

[1]F1-measure = 2 x precision x recall / (precision + recall)

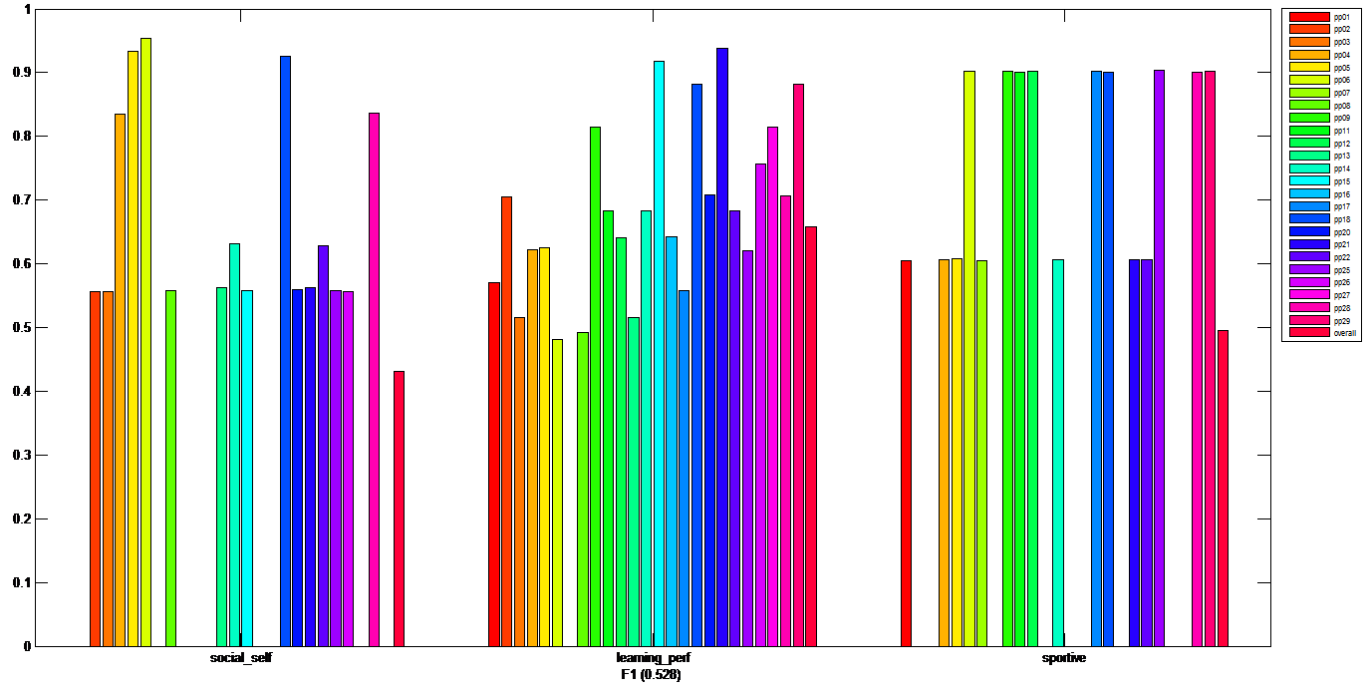

Figure 4. Model accuracy (i.e. F1-measure) per driver type for all participants with overall mean.

expert knowledge based on real-world data, by encoding the knowledge first and then training it on the new data, effectively combining human expertise and statistical data analysis in a uniform model for online learning and reasoning.

VI. Conclusions and Future Work

The Neural-Symbolic Cognitive Agent (NSCA) model and architecture presented in this paper offers an unified model capable of online learning, reasoning, and dynamic adaptation in complex real-world applications. The increasing need for multi-agent models and systems capable of learning and adaptation in dynamic environments demand novel learning algorithms, methods and tools. Our approach allows the modelled agents to learn rules about observed data in complex, data-intensive real-world scenarios (e.g. expert behaviour for training and assessment, visual intelligence, and driver type detection). Learned behaviour can be extracted to update existing domain knowledge for validation, reporting, maintenance, evolution and feedback. Furthermore the approach allows domain knowledge to be encoded in the model and deals with uncertainty in real-world data. The results described in this paper show that the NSCA is a generic agent model that can be applied to various domains, including Intelligent Transport Systems. In this domain the NSCA has proven to be a useful tool for the modelling of complex driving behaviour and driver type detection to support the reduction of CO_2 emissions. Part of future work will be the improvement of the driver type detection models in EcoDriver, by further training the NSCAs in a larger field test and validation of the encoded knowledge by human experts.

In summary, the NSCA provides an integrated model for learning, knowledge representation and reasoning capable of producing a realistic computational cognitive agent model. The NSCA seeks to address the challenge put forward in [4], [9], and contributes to the development of algorithms and tools for multi-agent deep learning and adaptation in dynamic and non-stationary environments.

References

[1] C. Fischer, "Feedback on household electricity consumption: a tool for saving energy?" *Energy Efficiency*, vol. 1, pp. 79–104, 2008.

[2] H. Alcott and S. Mullainathan, "Behavior and energy policy," *Science*, vol. 327, pp. 1204–1205, 2010.

[3] P. Stern, "Contributions of psychology to limiting climate change." *American Psychologist*, vol. 66, no. 4, pp. 303–3014, 2011.

[4] L. G. Valiant, "Three problems in computer science," *Journal of the ACM (JACM)*, vol. 50, no. 1, pp. 96–99, 2003.

[5] R. Borges, A. S. d'Avila Garcez, and L. C. Lamb, "Learning and Representing Temporal Knowledge in Recurrent Networks," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2409–2421, 2011.

[6] L. C. Lamb, R. Borges, and A. S. d'Avila Garcez, "A connectionist cognitive model for temporal synchronisation and learning," in *Proc. of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2007, pp. 827–832.

[7] L. de Penning, A. S. d'Avila Garcez, L. C. Lamb, and J.-J. C. Meyer, "A Neural-Symbolic Cognitive Agent for Online Learning and Reasoning," in *International Joint Conference on Artificial Intelligence*, Barcelona, Spain, 2011, pp. 1653–1658.

[8] A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay, *Neural-Symbolic Cognitive Reasoning*. Springer-Verlag New York Inc, 2009.

[9] M. Wooldridge, *An introduction to multiagent systems*, 2nd ed. Wiley, 2008.

[10] J. Lehmann, S. Bader, and P. Hitzler, "Extracting reduced logic programs from artificial neural networks," *Applied Intelligence*, vol. 32, no. 3, pp. 249–266, 2010.

[11] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," in *Parallel Distributed Processing: Volume 1: Foundations*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, vol. 1, pp. 194–281.

[12] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

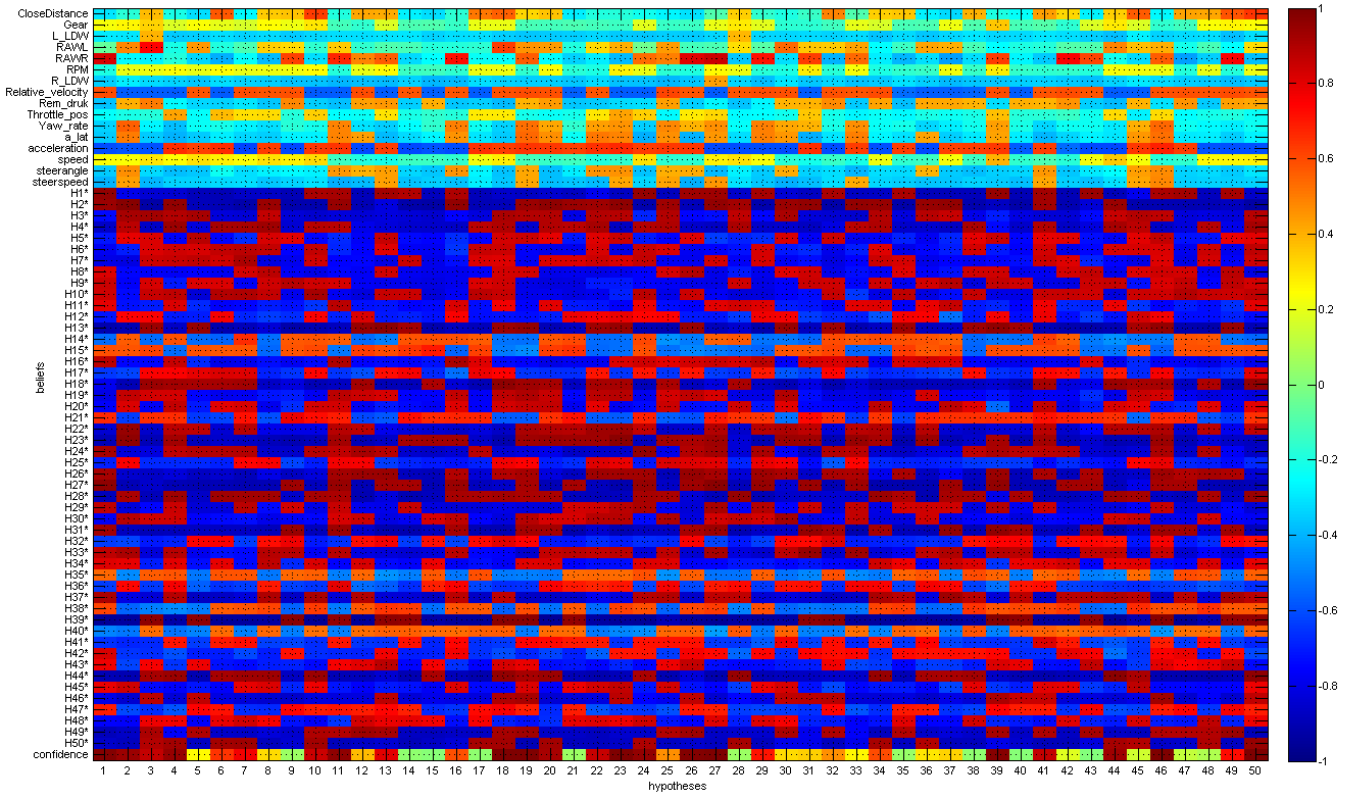

Figure 5. Visual representation of all temporal relations between hypotheses on driving behaviour and vehicle data, where the vertical axis depicts the beliefs on vehicle data and temporal relations $\bullet H$, and horizontal axis depicts the hypotheses H, where red denotes a high positive correlation and blue denotes a high negative correlation. Bottom row depicts the confidence c of the hypotheses (i.e. Eq. 6).

[13] I. Sutskever, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[14] A. Heuvelink, "Cognitive Models for Training Simulations," Ph.D. dissertation, Vrije Universiteit Amsterdam, Vrije Universiteit Amsterdam, 2009.

[15] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[16] M. Fisher, D. M. Gabbay, and L. Vila, *Handbook of temporal reasoning in artificial intelligence*. Elsevier, 2005.

[17] G. Pinkas, "Artificial Intelligence Reasoning , nonmonotonicity and learning in connectionist networks that capture propositional knowledge," *Artificial Intelligence*, vol. 77, pp. 203–247, 1995.

[18] P. M. Lee, *Bayesian statistics: an introduction*. John Wiley & Sons, 2012.

[19] L. de Penning, R. J. den Hollander, H. Bouma, G. J. Burghouts, and A. S. d'Avila Garcez, "A Neural-Symbolic Cognitive Agent with a Mind's Eye," in *Workshop on Neural-Symbolic Learning and Reasoning at AAAI*, Toronto, 2012.

[20] S. Choi, J. Kim, and D. Kwak, "Analysis and classification of driver behavior using in-vehicle can-bus information," in *Biennial Workshop on DSP for In-Vehicle and Mobile Systems*, 2007.

[21] M. Lu, J. Wang, K. Li, T. Yamamura, and N. Kuge, "Classification of Longitudinal Driving Behaviour based on Simulator Study," in *International Co-operation on Theories and Concepts in Traffic Safety (ICTCT)*, Leeds, UK, 2009, pp. 1–11.