EPiC Series in Computing

Volume 57, 2018, Pages 1–17

LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Verification, Testing, and Runtime Monitoring of Automotive Exhaust Emissions

Holger Hermanns[1], Sebastian Biewer[1], Pedro R. D'Argenio[2,3,1], and Maximilian A. Köhl[1]

[1] Saarland University, Saarland Informatics Campus, Germany
[2] Universidad Nacional de Córdoba, Argentina
[3] CONICET, Argentina

Abstract

Emission cleaning in modern cars is controlled by embedded software. In this context, the diesel emission scandal has made it apparent that the automotive industry is susceptible to fraudulent behaviour, implemented and effectuated by that control software. Mass effects make the individual controllers altogether have statistically significant adverse effects on people's health. This paper surveys recent work on the use of rigorous formal techniques to attack this problem. It starts off with an introduction into the dimension and facets of the problem from a software technology perspective. It then details approaches to use (i) model checking for the white-box analysis of the embedded software, (ii) model-based black-box testing to detect fraudulent behaviour under standardized conditions, and (iii) synthesis of runtime monitors for real driving emissions of cars in-the-wild. All these efforts aim at finding ways to eventually ban the problem of *doped* software, that is, of software that surreptitiously alters its behaviour in certain circumstances – against the interest of the owner or of society.

1 Introduction

Program verification and testing are methods for software manufacturers to check if their products satisfy certain objectives. Classically, these objectives agree with those of the users or the general interest. However, we observe a trend where the interests of the manufacturers diverge from the general interest, in particular in the context of embedded and cyber-physical systems. If the software includes functionality that is in the mere interest of the manufacturer, we call this software being *doped* [1].

Examples of software doping include customer lock-in strategies as found for instance in inkjet printers that refuse to work [2] when supplied with a toner or ink cartridge of a third party manufacturer despite technical compatibility. It is found in laptops and in electric vehicles that refuse to charge the battery if connected to a third-party charger [3, 4].

The diesel emissions scandal received a lot of attention [5] and is another example of software doping. Modern cars need to comply to a range of environmental regulations limiting the level of emissions for various toxic substances, greenhouse gases, and particles. The prime approach

to assure compliance with these regulations is black-box testing carried out in a controlled environment: Emission tests are carried out on a chassis dynamometer where the car is fixed but tires can rotate freely. During the test, emissions are measured at the exhaust pipe while the vehicle is made to follow a precisely defined profile meant to imitate real driving conditions. The conditions of the test, including speed profile and other details such as outside temperature, are both standardized and public, ensuring that the testing can be carried out in a reproducible way by an independent party, treating the car itself as a black box.

However, the singularity of the conditions on the chassis dynamometer makes it possible to infer when a car is undergoing an emission test and to intentionally adjust the car behaviour so as to comply with emission standards, while exceeding them during normal driving in favour of more economic resource usage. This surreptitious alteration of functionality is at the heart of the *diesel emissions scandal*, and is considered to be part of a component called *defeat device*. The US Environmental Protection Agency (EPA) defines a defeat device as a device that reduces the effectiveness of an emission control system under non-exceptional conditions [6].

A defeat device may use environmental parameters such as vehicle speed, travelled distance, time, horizontal acceleration, temperature, and more. The first defeat devices were discovered by EPA in September 2015 inside several Volkswagen and Audi cars [7]. Later, defeat devices were also found in cars made by Fiat [8, 9], Porsche [10] and Nissan [11]. Moreover, there have been allegations against several manufacturers, including Daimler [12], Peugeot [13], Renault [14], Citroën [15] and General Motors [16]. For some of them, it is as yet unclear if the emission thresholds were exceeded because of a defeat device, an incorrectly programmed software or insufficient emission cleaning hardware. Some BMW cars were, according to the manufacturer, temporarily affected due to a software bug, which increased emissions on the road *and* the chassis dynamometer [17, 18]. Other manufacturers cheated during the test execution instead of using defeat devices, e.g. Nissan [19], Mitsubishi [20], Mazda and Suzuki [21]. Opel has been cleared from prior allegations [22], but new allegations surfaced very recently [23].

In our work we focus on the exhaust gases NO and NO_2 (abbreviated as NO_x), but the formal methods we use can also be applied to other substances like particulate matter. The extrusion of NO_x gas is particularly a problem of diesel engines [24]. Roughly speaking, the problem in diesel engines is that the engine controller has limited control over the air-fuel mixture during combustion. Too much fuel causes a "rich" mixture that results an incomplete combustion and causes, among others, the engine to run hot. A surplus of air (respectively oxygen) gives a "lean" mixture instead, leading to additional reactions after the intended combustion, which results in an increased extrusion of NO_x gases. Three main technologies are available for the purpose of emissions cleaning systems: *exhaust gas recirculation* (EGR), *lean NO_x trapping* (LNT) and *selective catalyst reduction* (SCR), and are used in modern cars in different combinations to reduce the amount of extruded NO_x.

- EGR works by recirculating a portion of an engine's exhaust gas back to the engine cylinders. For this the engine controller operates a valve connecting the engine exhaust with the air intake. Opening the valve dilutes the oxygen in the intake air and thereby the temperature in the combustion chamber. Both effects lead to a reduction of the amount of NO_x eventually extruded.

- LNT uses a dedicated hardware catalyst positioned in the exhaust pipe. For LNT, the engine controller has to periodically alternate engine operation between lean and rich conditions. During lean operation, the catalyst adsorbs ("traps") NO_x contained in the exhaust to form nitrate species on the surface of the catalyst. Once these begin to saturate the catalyst surface, NO_x trapping efficiency begins to deteriorate, and the catalyst must

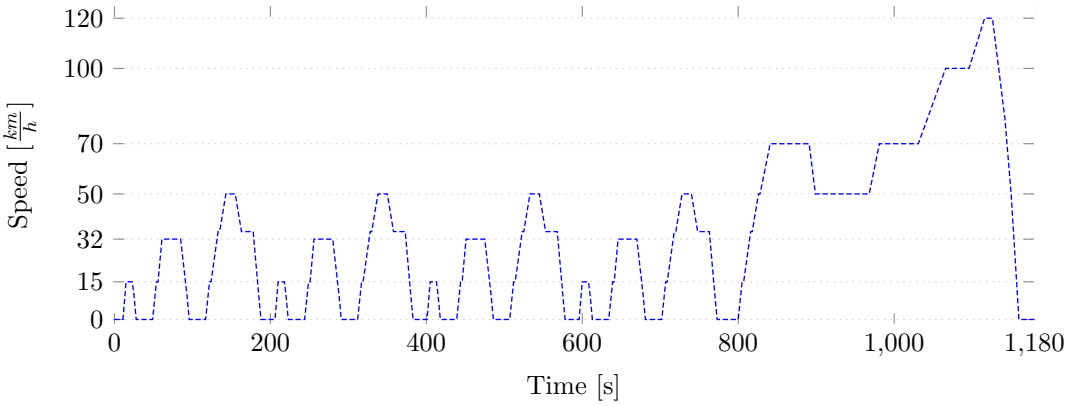

Figure 1: New European Driving Cycle (NEDC).

be "regenerated". For this, the nitrate is periodically burnt off while running the engine with a rich mixture. After regeneration the lean-rich cyclic process begins again.

- SCR is effectuated by the engine controller using dedicated hardware in the exhaust pipe, too. SCR effectuates a chemical reactions in the exhaust to reduce the amount of NO_x. For this, urea is injected into the exhaust stream and there reacts with NO_x and oxygen, resulting in nitrogen, carbon-dioxide and water. The urea liquid (called AddBlue) is stored in a tank that needs to be refilled just like the car's fuel tank in order to keep SCR effective.

There are various trade-offs between engine performance, engine wear, and emissions cleaning efficacy for each of the above technologies. SCR is commonly accepted as the most versatile and effective technology, but comes with the need to refill a second liquid.

As discussed above, cars have to pass various precisely defined tests prior to entering the market. Common to all regulations is a standardized test cycle that defines a speed profile that has to be followed on a chassis dynamometer. Other environmental parameters, like ambient air temperature, chassis inertia, etc. are also defined. This procedure allows for comparability between different car models and different manufacturers. Figure 1 shows the *New European Driving Cycle* (NEDC), which was the standard cycle until the emission scandal was discovered. It has been replaced by the *Worldwide harmonized Light vehicles Test Procedure* (WLTP) in 2017 in Europe, with the intention to cover more realistic driving behaviour than with the NEDC. Furthermore the European Union decided to gradually introduce *Real-Driving Emissions* (RDE) in their emissions regulation. RDE defines a framework for test cycles to be performed on the road instead of on a chassis dynamometer.

Since the revelation of the scandal, several scientific works have discussed the problem of defeat devices in diesel-powered cars, and how to detect them. We will review them in this paper. The published work ranges from attacking concrete car models to more conceptual approaches applicable to any model of any manufacturer. Some of them assume a white-box setting, i.e., they assume access to the engine controller code to perform a formal analysis, while black-box approaches do not. The latter, however, suffer from the significant effort needed to interact physically with real cars with human drivers in-the-loop.

We discuss altogether four approaches. We start off in Section 2 by reviewing the work of

Domke et al., who provided a very detailed and insightful analysis of Volkswagen and Fiat cases. This is a white-box approach where the authors acquired firmware images and documentation of an Engine Control Unit (ECU) fabricated by supplier Bosch. We then discuss how their observations can be generalised, leading to the definition of *robust cleanness* [25]. Any car model can be checked for being robustly clean, as described in Section 3, using hyperproperty model checking. However, this work assumes a white-box setting, too, which makes it limited in practical applicability, due to intellectual property restrictions in the automotive domain. Section 4 therefore explains how the definition of robust cleanness can be twisted to support black-box testing of real cars in connection with standardized test procedures [26]. Section 5 presents a runtime verification approach [27] to monitor how often regular trips driven in-the-wild satisfy the conditions of the RDE regulations and to check whether a trip passes the regulation. Section 6 summarises the paper with a discussion of the greater context.

2 White-Box Analysis of Defeat Devices

The diesel emissions scandal started in the US in 2015 when certain Volkswagen models were identified as exceeding emission limits systematically. Domke et al. have investigated how the defeat devices in Volkswagen cars work [9]. All ECUs considered by them are supplied by Bosch. The authors have used firmware images, their function sheets and other supplemental material they obtained from an official VW repair portal or from the chip-tuning community. Their findings revolve around code that determines whether a "customer-specific acoustic condition" is enabled. Notably, the "customer" is Bosch's customer Volkswagen. The cleaning technologies used are EGR and SCR.

Acoustics in engine control. The "acoustic condition" has several activation conditions that span different temperature and atmospheric pressure ranges. These conditions are easily met for testing and real-world conditions. In addition, there are deactivating conditions. The most interesting deactivating condition was added to some ECUs in 2007. There, Bosch introduced a measurement for the travelled distance d since engine start together with the time t since the vehicle first exceeds a configurable velocity. These ECUs also allow the car manufacturer to configure sets of values, which internally define curves by linearly interpolating the values of each set, as functions of time t. Curves are defined in pairs d_{lower} and d_{upper}, that together define a *profile*. When the car is used, the trip defines an additional curve d_{trip}. The current trip is said to *get outside* a profile, if and only if at some point in time t the value of $d_{trip}(t)$ is below the value $d_{lower}(t)$ or above $d_{upper}(t)$. There can be several profiles encoded in the ECU. If the trip gets outside of each profile at least once, the "acoustic condition" is cancelled and remains cancelled until engine restart.

Profiles as defeat mechanisms. The profiles found in several VW cars precisely describe some of the standard emission test cycles. Figure 2 shows three profiles that were found in a VW Sharan built in 2013 [28]. The white regions define the profile, i.e. for curves that always stay in one of the white regions the "acoustic condition" remains enabled. The blue curve in that figure shows the distance travelled when driving the NEDC (cf. Figure 1), which was the legally binding standard cycle when this car was admitted. It is rather apparent that the bottom profile is designed to match the NEDC. The affected ECUs have two different modes of emission cleaning, where one is far more effective (and uses far more AdBlue and/or induces more engine wear and less fuel efficiency) than the other. Which of these modes is used, depends on the value of the "acoustic condition".

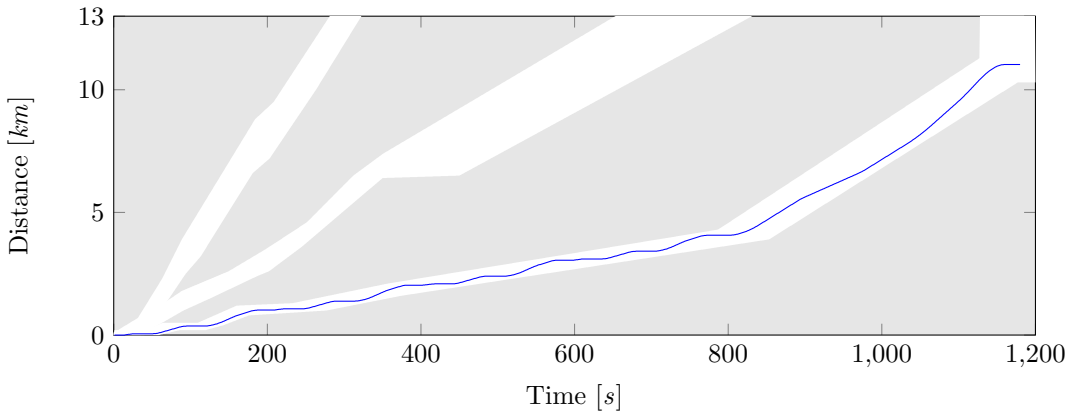

Figure 2: Three profiles found in a VW Sharan 2013. Trips leaving the white regions cancel the acoustic condition. The blue curve shows the distance profile of the NEDC.

The apparent intention of this functionality is to detect whether or not the car runs a standard emission test cycle, and to adjust the emission cleaning accordingly. Under test conditions, the "acoustic condition" remains enabled, while if run outside the standard emission test cycle, as it would normally happen while on the road, the "acoustic condition" would be deactivated. There is no obvious reason why this condition was given the name "acoustic condition" (by Bosch). With parameters as set by Volkswagen it is a defeat device: Proper emission cleaning is activated only if the "acoustic condition" is enabled.

Static code analysis for detecting "acoustic functions". In order to detect this kind of deactivation conditions in a large number of firmware images, the authors of [9] developed a prototype tool based on static analysis called *CurveDiff*. They analysed 963 firmware images and found defeat devices in 333 of them.

3 Clean and Doped Software

The entire diesel emissions scandal revolves around defeat devices. A defeat device is part of a controller or its software and provides functionality that is intended by the manufacturer to exist, but it is neither advertised with the product nor desired by the user or by society. It is added intentionally by the software manufacturer, and this is what disambiguates this phenomenon from a classical software (or hardware) bug. The nondisclosure of this questionable functionality is what leads to the suspicion of fraudulence. Software containing a defeat device matches the (informal) definition of *software doping* [1]:

> A software system is doped if the manufacturer has included a hidden functionality in such a way that the resulting behaviour intentionally favours a designated party, against the interest of society or of the software licensee.

The most prominent examples within the diesel emissions scandal are the sophisticated defeat devices deployed by Volkswagen, as discussed right above. Due to the tightly engineered pairs of piecewise linear functions, a small change in input (i.e., speed) may turn off emission cleaning

and thus increase the amount of extruded NO_x significantly. For example, when driving the NEDC 5km/h slower, the car discussed in Figure 2 deactivates the acoustic condition within less than three minutes. Intuitively, we would expect from a software that if the inputs deviate only slightly, then the output also does not change dramatically. In other words, we would expect the function representing the behaviour to be "robust" to the supplied inputs (i.e., the function's value should not change drastically for inputs that are different, but close).

Robust cleanness. The *robust cleanness* property can be defined formally [25]. The software is assumed to be a non-deterministic reactive program $P : \mathsf{In}^\omega \to 2^{(\mathsf{Out}^\omega)}$. P is defined on an infinite input sequence $i \in \mathsf{In}^\omega$ and reacts on the k-th input in this sequence producing the k-th output in each respective output sequence $o \in \mathsf{Out}^\omega$. The concrete sets In and Out depend on the context. For the diesel problem, a suitable input domain are vectors of all sensor data of the car and for the output domain the amount of NO_x in the exhaust. We further assume a set $\mathsf{StdIn} \subseteq \mathsf{In}^\omega$ of standard inputs, which could be, for example, the input trace for a standard driving cycle. For having control about how relevant a sensor input is for comparison of two inputs, we have to define a distance function $d_{\mathsf{In}} : (\mathsf{In}^* \times \mathsf{In}^*) \to \mathbb{R}_{\geq 0}$ between two (finite) input prefixes. Then we can limit the relevance of the change on the input value by some real-valued bound κ_{i}. In order to check the output, we assume a distance function $d_{\mathsf{Out}} : (\mathsf{Out} \times \mathsf{Out}) \to \mathbb{R}_{\geq 0}$ for single output symbols. A bound κ_{o} is used to decide whether a change in output is still reasonable. The value domains In and Out, the distance functions d_{In} and d_{Out}, their thresholds κ_{i} and κ_{o} and the standard inputs StdIn must be fixed in a contract between the manufacturer and the user. In the future, some of these contract parameters could also be defined as part of the emissions regulations.

Definition 1 checks for every standard input sequence prefix and every input sequence prefix with a maximum distance of κ_{i} to this standard, whether the two output sets have a distance of at most κ_{o}. For checking the distance of output sets, we use the Hausdorff distance, which is rolled out to the two sub-conditions in Definition 1.

Definition 1. *A non-deterministic reactive program P is* robustly clean[1] *if for all standard input sequences $i \in \mathsf{StdIn}$ and input sequences $i \in \mathsf{In}^\omega$ it holds that for all $k \geq 0$ and given that $\forall j \leq k : d_{\mathsf{In}}(i[..j], i'[..j]) \leq \kappa_{\mathsf{i}}$:*

1. *for every $o \in P(i)$, there exists $o' \in P(i')$ s.t. $d_{\mathsf{Out}}(o[k], o'[k]) \leq \kappa_{\mathsf{o}}$ and*
2. *for every $o' \in P(i')$, there exists $o \in P(i)$ s.t. $d_{\mathsf{Out}}(o[k], o'[k]) \leq \kappa_{\mathsf{o}}$.*

For $k \in \mathbb{N}$, we let $\sigma[k]$ and $\sigma[..k]$ denote respectively the k-th element of σ and the k-th prefix of σ.

Formalisation in HyperLTL. Provided a model of an emission cleaning (or ECU) is available, Definition 1 can be verified or refuted formally. Robust cleanness is a 2-safety hyperproperty, so classical temporal logics on traces are not suitable. HyperLTL is an extension of linear-time temporal logic (LTL) with trace quantifiers and trace variables [29, 30]. It allows to argue about multiple execution traces at the same time. A HyperLTL formula is defined by the following grammar:

$$\begin{array}{lcl} \psi & ::= & \exists \pi.\, \psi \mid \forall \pi.\, \psi \mid \phi \\ \phi & ::= & a_\pi \mid \neg\phi \mid \phi \vee \phi \mid \mathsf{X}\,\phi \mid \phi\,\mathsf{U}\,\phi \end{array}$$

[1] In contrast to [25] we do not explicitly consider parameters (as they can be simulated by inputs), we assume past-forgetful output distances (for presentation purposes) and we roll out the Hausdorff distance.

The quantifiers $\exists$ and $\forall$ quantify existentially and universally, respectively, over the set of traces. For example, the formula $\forall\pi.\,\exists\pi'.\,\phi$ means that for every trace π there exists another trace π' such that ϕ holds over the pair of traces. In order to refer to the values of the atomic propositions in the different traces, the atomic propositions are indexed with trace variables: for some atomic proposition $a \in \mathsf{AP}$ and some trace variable $\pi \in \mathcal{V}$, a_π states that a holds in the initial position of trace π. The temporal operators and Boolean connectives are interpreted as usual. In particular, $\mathsf{X}\,\phi$ means that ϕ holds in the next state of every trace under consideration. Likewise, $\phi\,\mathsf{U}\,\phi'$ means that ϕ' eventually holds in every trace under consideration at the same point in time, provided ϕ holds in every previous instant in all such traces. We also use the standard derived operators: $\mathsf{F}\,\phi \equiv \mathsf{true}\,\mathsf{U}\,\phi$, $\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$, and $\phi\,\mathsf{W}\,\phi' \equiv \neg(\neg\phi'\,\mathsf{U}\,(\neg\phi \wedge \neg\phi'))$.

We refer to [29] for the formal semantics of HyperLTL. To formalise robust cleanness in HyperLTL, we define the set of atomic propositions $\mathsf{AP} = \mathsf{AP}_{\mathsf{In}} \cup \mathsf{AP}_{\mathsf{Out}}$ to be the union of atomic propositions defining the inputs and atomic propositions defining the output. Hence, we have $\mathsf{In} = 2^{\mathsf{AP}_{\mathsf{In}}}$ and $\mathsf{Out} = 2^{\mathsf{AP}_{\mathsf{Out}}}$ and the program $P \subseteq (2^{\mathsf{AP}})^\omega$ can be seen as a function $\hat{P} : \mathsf{In}^\omega \to 2^{(\mathsf{Out}^\omega)}$ where

$$t \in P \quad \text{if and only if} \quad (t \downarrow \mathsf{AP}_{\mathsf{Out}}) \;\in\; \hat{P}(t \downarrow \mathsf{AP}_{\mathsf{In}}),$$

with $t \downarrow A$ defined by $(t \downarrow A)[k] = t[k] \cap A$ for all $k \in \mathbb{N}$.

In the following, we assume a past-forgetful input distance function, i.e. the distance only depends on the last element of the input traces. In the HyperLTL formulas, we will use the function $\hat{d}_{\mathsf{In}} : (\mathsf{In} \times \mathsf{In}) \to \mathbb{R}_{\geq 0}$ such that $d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}') = \hat{d}_{\mathsf{In}}(\mathsf{last}(\mathsf{i}), \mathsf{last}(\mathsf{i}'))$ for every $\mathsf{i}, \mathsf{i}' \in \mathsf{In}^*$. The output distance function is past-forgetful already by definition. We use syntactic sugar and let the trace predicate StdIn_π hold if and only if $\pi \downarrow \mathsf{AP}_{\mathsf{In}}$ is a standard input sequence. More syntactic sugar is used for checking value equality and for the checks if the input or output distance is below its threshold: $\mathsf{i}_\pi = \mathsf{i}_{\pi'}$, $\mathsf{o}_\pi = \mathsf{o}_{\pi'}$, $\hat{d}_{\mathsf{In}}(\mathsf{i}_\pi, \mathsf{i}_{\pi'}) \leq \kappa_{\mathsf{i}}$ and $d_{\mathsf{Out}}(\mathsf{o}_\pi, \mathsf{o}_{\pi'}) \leq \kappa_{\mathsf{o}}$. Proposition 1 spells out the HyperLTL formulas for both parts of the robust cleanness definition (Definition 1.1 and 1.2).

Proposition 1. *A reactive program P is robustly clean under past-forgetful distances d_{In} and d_{Out} if and only if P satisfies the following two HyperLTL formulas*

$$\begin{aligned}
&\forall\pi_1.\,\forall\pi_2.\,\exists\pi_2'.\\
&\quad \mathsf{StdIn}_{\pi_1} \rightarrow \Big(\mathsf{G}(\mathsf{i}_{\pi_2} = \mathsf{i}_{\pi_2'}) \wedge \big((d_{\mathsf{Out}}(\mathsf{o}_{\pi_1}, \mathsf{o}_{\pi_2'}) \leq \kappa_{\mathsf{o}})\;\mathsf{W}\;(\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1}, \mathsf{i}_{\pi_2'}) > \kappa_{\mathsf{i}})\big)\Big)\\
&\forall\pi_1.\,\forall\pi_2.\,\exists\pi_1'.\\
&\quad \mathsf{StdIn}_{\pi_1} \rightarrow \Big(\mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_1'}) \wedge \big((d_{\mathsf{Out}}(\mathsf{o}_{\pi_1'}, \mathsf{o}_{\pi_2}) \leq \kappa_{\mathsf{o}})\;\mathsf{W}\;(\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1'}, \mathsf{i}_{\pi_2}) > \kappa_{\mathsf{i}})\big)\Big)
\end{aligned}$$

The correctness proof for the above proposition echoes the proof of Proposition 19 in [25].

An example application. We verified a toy program with respect to the question whether it is robustly clean. We refuted the same property for a doped variation of that program using the HyperLTL model-checker MCHyper [30]. In its current version, MCHyper so far accepts formulas that either have only universal quantifiers or existential quantifiers, but not both. So, we needed to modify formulas and use a particular proof technique to show/refute robust cleanness, see [25].

4 Doping Tests

The techniques presented in Sections 2 and 3 assume that the implementation of the engine control unit is available, for example as a firmware image or in terms of a formal model. This assumption is not realistic in the context of the diesel emissions scandal, because automotive embedded control software is usually a proprietary code and therefore rarely available openly. Instead, we are unfortunately forced to view the car as a black box. Common practice for black-box analysis is testing. This section sketches the formal foundations [26] for model-based black-box doping tests and demonstrates its application.

Model-based testing. We use the theory of model-based testing [31, 32] for our approach. It has originally been proposed for specifications that are modelled as labelled transition systems (LTS). The definition below defines LTS and input-output transition systems (IOTS). We assume that the *system under test* (SUT), i.e., the car, can be modelled as an IOTS. For ease of presentation, we do not consider internal transitions (τ).

Definition 2. *A* labelled transition system (LTS) *with inputs and outputs is a tuple* $\langle Q, \mathsf{In}, \mathsf{Out}, \to, q_0 \rangle$ *where* Q *is a (possibly uncountable) non-empty set of states,* $L = \mathsf{In} \uplus \mathsf{Out}$ *is a (possibly uncountable) set of labels,* $\to \subseteq Q \times L \times Q$ *is the transition relation, and* $q_0 \in Q$ *is the initial state.*

We say that a LTS is an input-output transition system (IOTS) *if all its input actions are enabled in any state, i.e., for all* $s \in Q$ *and* $a \in \mathsf{In}$ *there is some* $s' \in Q$ *such that* $s \xrightarrow{a} s'$.

A *finite path* p in an LTS $\mathcal{L}$ is a sequence $s_1 a_1 s_2 a_2 \dots a_{n-1} s_n$ with $s_i \xrightarrow{a_i} s_{i+1}$ for all $1 \leq i < n$. Similarly, an *infinite path* p in $\mathcal{L}$ is a sequence $s_1 a_1 s_2 a_2 \dots$ with $s_i \xrightarrow{a_i} s_{i+1}$ for all $i \in \mathbb{N}$. Let $\mathsf{paths}_*(\mathcal{L})$ and $\mathsf{paths}_\omega(\mathcal{L})$ be the sets of all finite and infinite paths of $\mathcal{L}$ beginning in the initial states, respectively. The sequence $a_1 a_2 \cdots a_n$ is a *finite trace* of $\mathcal{L}$ if there is a finite path $s_1 a_1 s_2 a_2 \dots a_{n-1} s_n \in \mathsf{paths}_*(\mathcal{L})$, and $a_1 a_2 \cdots$ is an *infinite trace* if there is an infinite path $s_1 a_1 s_2 a_2 \dots \in \mathsf{paths}_\omega(\mathcal{L})$. Let $\mathsf{traces}_*(\mathcal{L})$ and $\mathsf{traces}_\omega(\mathcal{L})$ be the sets of all finite and infinite traces of $\mathcal{L}$, respectively.

Model-based doping tests. To capture the notion of software doping in the setting of LTS, we provide two projections of a trace, one that projects to only input labels and another projecting to only output labels. To do so, we extend the set of labels by adding the input $-_i$, that indicates that in the respective step some output was produced (but masking the precise output), and the output $-_o$ that indicates that in this step some (masked) input was given. The *projection on inputs* $\downarrow_i : L^\omega \to (\mathsf{In} \cup \{-_i\})^\omega$ and the *projection on outputs* $\downarrow_o : L^\omega \to (\mathsf{Out} \cup \{-_o\})^\omega$ are defined for all traces σ and $k \in \mathbb{N}$ as follows, and lifted to sets of traces in the usual elementwise way.

$$\sigma{\downarrow_i}[k] \coloneqq \textbf{ if } \sigma[k] \in \mathsf{In} \textbf{ then } \sigma[k] \textbf{ else } -_i$$
$$\sigma{\downarrow_o}[k] \coloneqq \textbf{ if } \sigma[k] \in \mathsf{Out} \textbf{ then } \sigma[k] \textbf{ else } -_o$$

In this extended setting, we assume that the distance functions $d_{\mathsf{In}} : \big((\mathsf{In} \cup \{-_i\})^* \times (\mathsf{In} \cup \{-_i\})^*\big) \to \mathbb{R}_{\geq 0}$ and $d_{\mathsf{Out}} : \big((\mathsf{Out} \cup \{-_o\}) \times (\mathsf{Out} \cup \{-_o\})\big) \to \mathbb{R}_{\geq 0}$ run on traces containing labels $-_i$ and $-_o$. For testing, it is important to know the standard behaviour of the car at the time when the test result (**pass** or **fail**) is decided. Hence we enhance the set StdIn to a LTS $\mathcal{S}$ that reflects the standard inputs and their outputs of the program.

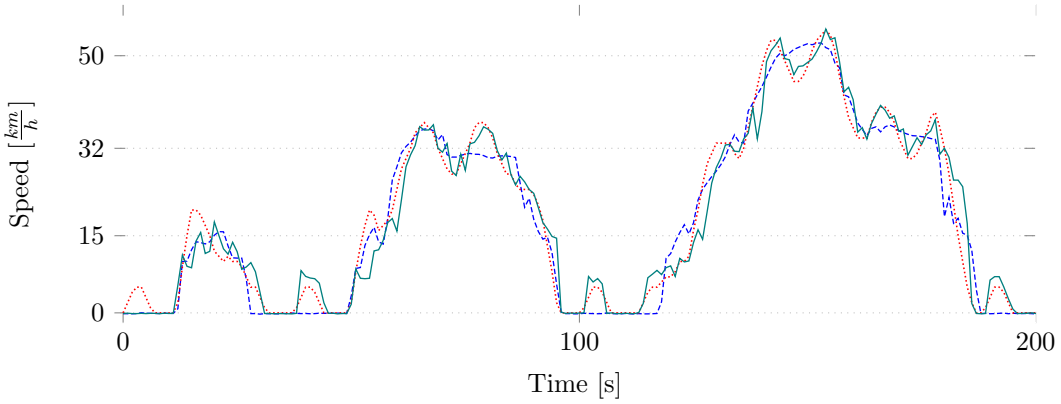

Figure 3: Initial 200s of a SineNEDC (red, dotted), its test drive (green) and the NEDC driven (blue, dashed).

Definition 3. *A LTS $\mathcal{S}$ is* standard *for a LTS $\mathcal{L}$, if for all $\sigma \in \mathsf{traces}_\omega(\mathcal{S})$ and $\sigma' \in \mathsf{traces}_\omega(\mathcal{L})$, $\sigma{\downarrow_i} = \sigma'{\downarrow_i}$ implies $\sigma' \in \mathsf{traces}_\omega(\mathcal{S})$.*

Now the definition of robustly clean can be restated in terms of LTS as follows.

Definition 4. *Let $\mathcal{L}$ be an LTS and $\mathcal{S}$ its standard LTS. $\mathcal{L}$ is* robustly clean *if for all $\sigma \in \mathsf{traces}_\omega(\mathcal{S})$ and $\sigma' \in \mathsf{traces}_\omega(\mathcal{L})$ such that then for all $k \geq 0$ such that $d_{\mathsf{In}}(\sigma[..j]{\downarrow_i}, \sigma'[..j]{\downarrow_i}) \leq \kappa_{\mathsf{i}}$ for all $j \leq k$, the following holds:*

1. *there exists $\sigma'' \in \mathsf{traces}_\omega(\mathcal{L})$ s.t. $\sigma'{\downarrow_i} = \sigma''{\downarrow_i}$ and $d_{\mathsf{Out}}(\sigma[k]{\downarrow_o}, \sigma''[k]{\downarrow_o}) \leq \kappa_{\mathsf{o}}$ and*
2. *there exists $\sigma'' \in \mathsf{traces}_\omega(\mathcal{L})$ s.t. $\sigma{\downarrow_i} = \sigma''{\downarrow_i}$ and $d_{\mathsf{Out}}(\sigma'[k]{\downarrow_o}, \sigma''[k]{\downarrow_o}) \leq \kappa_{\mathsf{o}}$.*

The above definition echoes the semantics of the HyperLTL interpretation appearing in Proposition 1. Thus, the proof showing that Definition 4 is the correct interpretation of Definition 1 in terms of LTS echoes that of Proposition 1.

Test generation. With these definitions, it is now possible [26] to develop an automatic test generation algorithm that produces sound and exhaustive test suites. Such a test suite can decide whether a software is clean: if it is clean, then all test cases of the suite will pass and otherwise, there is at least one test case that fails. This is, of course, only a theoretic result, because such a test suite may contain infinitely many test cases [31, 32].

A considerable open problem for doping test practicality is an efficient test case selection. There are test cases that are more likely to reveal software doping than others. Even more problematic is that our algorithm may produce test cases that are not executable, because they require the tester to drive the car in a way that is technically impossible (e.g. if demanding too drastic acceleration). We are actively investigating an automatic test generation approach, that produces realistic test cases and which resolves non-determinism during generation by picking the test case most promising for letting it fail. At the current stage, we hand-craft our test-cases. The execution of the test is done by a human that is controlling the car. This methodology comes with the problem, that, due to driving mistakes, the human driver will notoriously miss to pass the designated inputs to the car correctly. To overcome this problem,

Figure 4: Nissan NV200 Evalia on a dynamometer.

we use a monitoring approach: we record the execution of the designated test case and analyse the trip afterwards.

A concrete doping test case. The red dotted line in Figure 3 shows one of our hand-crafted test cycles. It is defined as an NEDC superimposed by a sine curve: $\textsc{SineNEDC}(t) \coloneqq \text{NEDC}(t) + 5 \cdot \sin(0.5t)$, where t is the time passed since the beginning of the test.

We fix a contract, where the input space In is a vector space spanned by all possible input parameter dimensions. We can use past-forgetful distances with $d_{\mathsf{In}}(\vec{a}, \vec{b}) \coloneqq |v(\vec{a}) - v(\vec{b})|$ if $\vec{a}, \vec{b} \in \mathsf{In}$, $d_{\mathsf{In}}(-_i, -_i) = 0$ and $d_{\mathsf{In}}(a, b) = \infty$ otherwise. For $\vec{a} \in \mathsf{In}$, we distinguish the speed dimension as $v(\vec{a}) \in \mathbb{R}$ (measured in km/h). We also take $\mathsf{Out} = \mathbb{R}$ for the average amount of NO_x gases per kilometre since engine start (in mg/km). The speed is the decisive quantity defined to vary along common test cycles (cf. Figure 1). Hence $d_{\mathsf{In}}(\vec{a}, \vec{b}) = 0$ if $v(\vec{a}) = v(\vec{b})$ regardless of the values of other parameters. We define $d_{\mathsf{Out}}(a, b) = |a - b|$ if $a, b \in \mathsf{Out}$, $d_{\mathsf{Out}}(a, b) = 0$ if $a, b \in \{-_o, \delta\}$, and $d_{\mathsf{Out}}(a, b) = \infty$ otherwise. We consider inputs where the maximum difference of speed are 15km/h, i.e. $\kappa_{\mathsf{i}} = 15km/h$. The European emissions regulation defines a threshold for NO_x of 80mg/km for modern cars. Still, we grant a generous NO_x difference $\kappa_{\mathsf{o}} = 180$mg/km.

We tested a Renault 1.5 dci ($110hp$) (Diesel) engine. This engine runs, among others, inside a Nissan NV200 Evalia which is classified as a Euro 6b car. The test cycle used in the original type approval of the car was NEDC. Emissions are cleaned using exhaust gas recirculation (EGR). The car was fixed on a *Maha LPS 2000* dynamometer with an *AVL M.O.V.E iS* portable emissions measurement system (PEMS) attached (see Figure 4) and data is sampled with a rate of 1 *Hz*. For reference, we drove the NEDC, which is depicted in the blue dashed line of Figure 3. The peak deviation of our NEDC drive to the regulation are 9 km/h. The execution of SineNEDC is depicted in the green solid line. The speed difference of the (actually executed) SineNEDC and NEDC was always less or equal to 14km/h, i.e. less or equal to κ_{i}. During the SineNEDC-drive, the car produced about 3.24 times the amount of NO_x, that is 404 mg/km more than what we measured for the NEDC, which is a violation of the contract, and effectively an indication of doped software.

	Urban	Rural	Motorway
Ratio Range [%]	$[29, 44]$	$[23, 43]$	$[23, 43]$
Speed Range [km/h]	$[0, 60]$	$]60, 90]$	$]90, 160]$
Distance [km]	≥ 16	≥ 16	≥ 16
Additional Constraints	stop percentage between 6% and 30% of urban time; average velocity in range $[15, 40]$km/h		$>$ 100km/h for at least 5mins
Temperature [K]	moderate: $[273, 303]$; extended: $[266, 273[$ or $]303, 308]$		
Altitude [m]	moderate: < 700; extended: $]700, 1300]$		
Speed Limit [km/h]	145 (]145,160] for at most 3% of motorway time)		

Table 1: Some constraints for the urban, rural and motorway phase of RDE tests.

5 Real Driving Emissions

The Real Driving Emissions (RDE) test procedure complements the Worldwide harmonized Light vehicles Test Procedure (WLTP) with on-the-road tests under presumably realistic conditions. While the WLTP is considered more realistic than previous procedures, it still shares their problematic characteristics in that tests need to be conducted on a chassis dynamometer and the driving profile is very much a singularity, making it easy to identify by doped control software.

RDE regulation and its execution. The RDE regulation instead comes with broad certification conditions for tests conducted under real-word conditions, on public roads and during working days. An (informal) specification document [33] spells out precise preconditions a trip, i.e., a trajectory driven with a car in-the-wild, has to satisfy to count as a valid RDE test. These preconditions include constraints on the route, allowed altitudes and speeds, and on the dynamics of the driving profile. An RDE test must traverse three phases, the urban, the rural, and the motorway phase covering different speed ranges and each making up approximately one third of the total trip distance. Table 1 provides an overview of the constraints for all three phases.

Whenever a trip meets the RDE preconditions, the emissions extruded during that trip must not have exceeded the respective emission limits. While performing an RDE, the relevant test data is obtained using a *Portable Emissions Measurement System* (PEMS). A PEMS comprises a small and lightweight emission laboratory which can be carried with the vehicle during the test. It is hooked up to the tailpipe of the vehicle to measure the emissions and to the Engine Control Unit (ECU) to obtain additional data like the current velocity. Usually, proprietary software like "AVL Concerto for PEMS" [34] is used to evaluate whether the driven trip indeed was a valid RDE trip and whether the emission limits were obeyed. Due to its nature proprietary software opens up another surface for manipulation as the source code is not available and, hence, cannot be inspected by the public.

Lola 2.0 and aggregation windows. We formalized the RDE regulation so as to make it accessible to runtime monitoring based on a formal model [27]. Runtime monitoring, in

Figure 5: Our prototype monitoring system (left) comprising an OBD-II Dongle, a GPS module, a CAN interface, a pressure and temperature sensor, an accelerometer, and a Raspberry Pi. The OBD-II dongle connected to the OBD-II Port of the Audi A7 (right).

contrast to traditional testing, has the advantage that actual systems in-use are supervised under real-word conditions. It therefore complements ahead-of-time techniques which are only able to cover individual systems under scrutiny. To arrive at our formalization we started off from the stream-based specification language and runtime monitoring framework Lola 2.0 [35, 36] as a basis, and extended that with sliding aggregation windows [27]. We refer to the publications mentioned here for details on Lola 2.0 and its extension, and only present an intuitive description here.

The sliding aggregation windows extension enables the succinct specification and efficient computation of percentiles and other aggregation values. For instance, the stream

```
output float va95 := percentile95((v * a)[-n:0 | a_is_positive])
```

computes the 95% percentile of speed values `v` times positive acceleration values `a` over the last `n` samples. Intuitively a sliding aggregation window applies the aggregation function to the sequence of values within the bounds of the window specifier for which the condition is satisfied. It turns out that sliding aggregation windows do not extend the expressiveness of Lola 2.0, since they can be encoded, at the price of size exacerbation, into ordinary syntax. For the full technical details on how to rewrite sliding aggregation windows and make use of efficient aggregation algorithms see [27].

RDE formalisation. With our extended version of Lola in place we were able to translate the RDE regulation into a Lola specification. Our openly available formalization[2] currently allows us to check whether a trip satisfies the RDE preconditions or, if the trip is longer than two hours, whether a two hour suffix does. A valid RDE trip must not last longer than two hours, so we only consider the maximal suffix in terms of trip duration. Thanks to an efficient rewriting of aggregation windows we are able to compute percentiles with an update cost of $\mathcal{O}(\log n)$ and $\mathcal{O}(n)$ memory where n is the width of the aggregation window. The RDE regulation requires that the driver neither drives too aggressive nor too restrained. To this end, it requires to compute the 95% percentile of speed times acceleration for acceleration values of at least 0.1 for each speed bin. We show this exemplary for the urban speed bin:

[2] See https://www.powver.org/real-driving-emissions/.

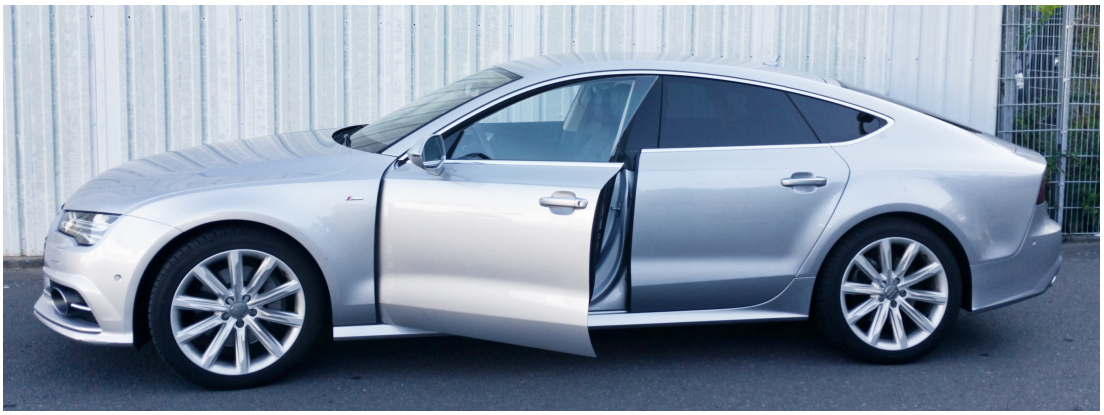

Figure 6: Audi A7 with an on-board NO_x sensor, prior to RDE testing.

```
output float a          := (v[+1,0] - v[-1,0]) / (2 * 3.6)
output float va         := (v * a / 3.6)
output bool  u_a_ge_01  := a >= 0.1 & is_urban
output float u_va_pct   := percentile95(va[-N:0 | u_a_ge_01])
```

In general, specifications with future references are not efficiently monitorable, i.e., they require a non-constant amount of memory [36]. However, as `v` is extended in every step and cannot delay the computation indefinitely, the future reference is indeed efficiently monitorable in this case.

Low-cost RDE monitoring. With the successful formalization of the RDE regulation, the Lola 2.0 framework can synthesize a runtime monitor. To validate our work, we first ran that monitor on recorded data of genuine RDE tests performed with a PEMS. The monitor behaved as expected and computed the results correctly. In addition, we developed a low-cost variant of the RDE monitoring procedure which does not require a PEMS, provided the car is equipped with some on-board exhaust emissions sensors. The key challenge is to obtain the relevant data to monitor—especially regarding the concentrations of the various regulated emission gasses. While the vehicle speed and altitude can be determined via GPS with an ordinary smartphone, measuring emissions requires specific sensors. Fortunately, many modern cars with an SCR system are already equipped with NO_x sensors measuring the NO_x concentration in the after-treatment exhaust stream, i.e., the stream of exhaust after it ran through the cleaning process as it leaves the tailpipe. These values can be read out via the standardized OBD-II interface [37]. Every modern car is indeed equipped with such an OBD-II standard debug port, which is accessible to the user. In our prototype, the monitor runs inside a Raspberry Pi, which is connected to a GPS module, a pressure and temperature sensor, an accelerometer, and a CAN/USB interface connecting to a OBD-II dongle. The hardware costs of that system are in the order of $100, and hence affordable by a layperson. This compares favourably to the costs of a PEMS system, which are in the order of $250,000.

A concrete runtime monitoring case. To put the low-cost monitor into concrete use, we embarked on using it for RDE tests with an Audi A7 3.0 TDI 200kW (see Figure 6). We conducted several of these RDE tests. The A7 3.0 TDI 200kW model is known to contain a

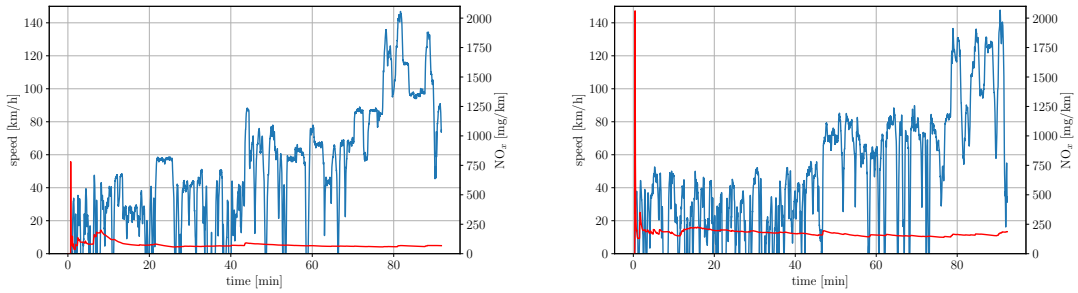

Figure 7: Profiles of low-cost RDE tests with an Audi A7 3.0 TDI 200kW with full (left) and almost empty (right) urea tank. The blue curve indicates the car velocity and the red curve shows the amount of extruded NO_x per kilometre.

defeat device which chokes the injected amount of urea shortly before running out of it [38]. It can be assumed that the car we inspected indeed conforms to the EURO 6 emission limits in case enough urea is in the tank, but if instead the urea tank is almost empty, NO_x emissions would exceed the thresholds. Indeed, the results of our tests with the low-cost RDE equipment and our monitor suggest that the car is doped. Figure 7 contains the results of two of our low-cost RDE test runs, one of them with a full urea tank and another one with an almost empty tank. For the former, the distance specific NO_x emissions converge towards 68 mg/km which is within the EURO 6 emission limit, even for tests performed on a chassis dynamometer (80 mg/km [39]). It almost matches the values from the data-sheet [40], underlining the appareant precision of our indirect NO_x concentration measurement. Assuming that the on-board NO_x sensor is not doped itself and delivers somewhat precise values, this is an excellent value for on-the-road driving. In contrast, in case of an almost empty tank, the NO_x emissions are considerably higher, they converge towards 187 mg/km which is 2.7 times as much as with a full tank. Assuming the effective EURO 6d-TEMP RDE emission limit of 168 mg/km for NO_x [41], this exceeds the emission limits and thus our monitor complained, just as expected. Notably, the approval of the car is based on EURO 6b [40] which only stipulates emission limits for the NEDC, so legally no emission limits apply for RDE tests. However, the drastic relative deviation between our test runs with full and with an almost empty urea tank suggests that there indeed is a defeat device in place in our particular vehicle—especially considering the knowledge how the model is doped [38].

Although research about the actual precision of on-board sensors is still in progress, our results show, that low-cost RDE monitoring and testing conducted by laypersons and using inexpensive equipment can provide interesting insights into real-world automotive exhaust emission. Due to the restrictive limits of EURO 6d it is likely that in the future more cars are equipped with SCR systems and with on-board NO_x sensors. Our monitor and low-cost RDE variant can then be utilized by the public to monitor emissions on a large scale.

6 Conclusion

This paper has provided a survey of recent and ongoing work focussing on formal techniques to identify defeat devices in automotive emissions cleaning. Apart from static analysis techniques targeting a particular (and sophisticated) defeat mechanism, we looked into formal ways to

state that an engine controller is robustly clean. We sketched how this property can be verified using hyperproperty model checking, and we explained how robust cleanness can be twisted to support black-box testing derived from standardized test procedures. Finally we discussed a runtime monitoring approach for the latest regulations regarding real driving emissions, RDE. We have put the last two contributions into practice, using black-box testing on a Euro 6 diesel-powered Nissan NV200 Evalia fixed to a chassis dynamometer, and using runtime monitoring on an Audi A7 equipped with a 3 litre Euro 6 diesel engine driving on normal roads. The latter uses low-cost infrastructure that potentially can be deployed massively to end customers for emissions monitoring, possibly combined with crowd-sourcing. In both concrete cases we were able to collect evidence that the emission cleaning mechanisms used within these cars are not acting as they should act.

Acknowledgements. We are grateful for inspiring discussions with and scientific contributions of Felix Domke, Gilles Barthe (IMDEA), Bernd Finkbeiner (Saarland University), as well as Peter Birtel, Michael Fries, and Thomas Heinze (htw saar). This work is partly supported by ERC Advanced Grant 695614 (POWVER), by the Saarbrücken Graduate School of Computer Science, by the Sino-German CDZ project 1023 (CAP), by ANPCyT PICT-2017-3894 (RAFTSys), and by SeCyT-UNC 33620180100354CB (ARES).

References

[1] G. Barthe, P. R. D'Argenio, B. Finkbeiner, and H. Hermanns, "Facets of software doping," in *7th International Symposium on Leveraging Applications of Formal Methods, ISOLA 2016, Part II*, ser. LNCS, vol. 9953. Springer, 2016, pp. 601–608.

[2] J. C. Dvorak, "The secret printer companies are keeping from you," PC Mag UK, http://uk.pcmag.com/printers/60628/opinion/the-secret-printer-companies-are-keeping-from-you, 2012, Online; accessed: 2018-03-23.

[3] E Bike Tuning, "BionX tuning," http://www.ebiketuning.com/comparison/bionx-tuning.html, Online; accessed: 2018-03-23.

[4] Tritech Computer Solutions, "Dell laptops reject third-party batteries and AC adapters/chargers. Hardware vendor lock-in?" https://nctritech.wordpress.com/2010/01/26/dell-laptops-reject-third-party-batteries-and-ac-adapterschargers-hardware-vendor-lock-in/, 2010, Online; accessed: 2018-03-23.

[5] Wikipedia, "Diesel emissions scandal," Wikipedia, The Free Encyclopedia, 2018, Online; accessed: 2018-10-16.

[6] US Code of Federal Regulations, "40 CFR §86." [Online]. Available: https://www.law.cornell.edu/cfr/text/40/86.1803-01

[7] United States Environmental Protection Agency , " Notice of Violation ." [Online]. Available: https://www.epa.gov/sites/production/files/2015-10/documents/vw-nov-caa-09-18-15.pdf

[8] J. Dunn, "Diesel emissions scandal: Fiat under investigation," The Telegraph, http://www.telegraph.co.uk/cars/news/diesel-emissions-scandal-fiat-under-investigation/, 2017, Online; accessed: 2018-10-16.

[9] M. Contag, G. Li, A. Pawlowski, F. Domke, K. Levchenko, T. Holz, and S. Savage, "How they did it: An analysis of emission defeat devices in modern automobiles," in *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2017, pp. 231–250.

[10] AFP, "Germany orders Porsche recall over diesel emissions cheating," phys.org, https://phys.org/news/2018-05-germany-porsche-recall-diesel-emissions.html, 2018, Online; accessed: 2018-10-09.

[11] J. Holder, "Nissan found guilty of using diesel emissions cheat device in South Korea," Autocar, https://www.autocar.co.uk/car-news/industry/

nissan-found-guilty-using-diesel-emissions-cheat-device-south-korea, 2017, Online; accessed: 2018-10-09.

[12] BBC, "Daimler forced to recall Mercedes with defeat devices," BBC, https://www.bbc.com/news/business-44444361, 2018, Online; accessed: 2018-10-09.

[13] AFP, "Peugeot suspected of fraud in diesel scandal," France 24, https://www.france24.com/en/20170908-peugeot-suspected-fraud-diesel-scandal-cars-france-business, 2017, Online; accessed: 2018-10-09.

[14] ——, "Renault 'cheated on 25 years of pollution tests'," The Local France, https://www.thelocal.fr/20170316/renault-cheated-on-25-years-of-pollution-tests, 2017, Online; accessed: 2018-10-09.

[15] Q. ARIÈS, "Citroën may have breached emissions rules: report," Politico, https://www.politico.eu/article/citroen-dieselgate-emissions/, 2017, Online; accessed: 2018-10-09.

[16] P. A. Eisenstein, "GM Accused of Cheating on Diesel Emissions," NBC News, https://www.nbcnews.com/business/autos/gm-accused-cheating-diesel-emissions-n765146, 2017, Online; accessed: 2018-10-09.

[17] K. Ott, "BMW soll zehn Millionen Euro Bußgeld zahlen," Süddeutsche Zeitung, https://www.sueddeutsche.de/wirtschaft/abgasreinigung-bmw-soll-zehn-millionen-euro-bussgeld-zahlen-1.4113929, 2018, Online; accessed: 2018-10-09.

[18] M. Hägler and K. Ott, "BMW im Abgasskandal vorerst entlastet," Süddeutsche Zeitung, https://www.sueddeutsche.de/wirtschaft/autoindustrie-bmw-im-abgasskandal-vorerst-entlastet-1.4077669, 2018, Online; accessed: 2018-10-09.

[19] BBC, "Nissan admits falsifying emissions tests in Japan," BBC News, https://www.bbc.com/news/business-44763905, 2018, Online; accessed: 2018-10-09.

[20] H. Sheffield, "Mitsubishi admits employees falsified fuel economy tests data," Independent, https://www.independent.co.uk/news/business/news/mitsubishi-says-cars-failed-fuel-tests-after-employees-falsified-emissions-data-a6992291.html, 2016, Online; accessed: 2018-10-09.

[21] N. YAMAMOTO, "Suzuki and Mazda admit faking emissions tests," Nikkei Asian Review, https://asia.nikkei.com/Business/Companies/Suzuki-and-Mazda-admit-faking-emissions-tests, 2018, Online; accessed: 2018-10-09.

[22] L. Allan, "Vauxhall and Opel cleared of emissions tampering," Auto Express, https://www.autoexpress.co.uk/fiat/95558/vauxhall-and-opel-cleared-of-emissions-tampering, 2017, Online; accessed: 2018-10-09.

[23] DPA, "Diesel-Razzia bei Opel," Süddeutsche Zeitung, https://www.sueddeutsche.de/wirtschaft/eil-diesel-razzia-bei-opel-1.4170707, 2018, Online; accessed: 2018-10-16.

[24] The International Council on Clean Transportation. (2015) Nox control technologies for Euro 6 diesel passenger cars: Market penetration and experimental performance assessment. [Online]. Available: http://theicct.org/sites/default/files/publications/ICCT_NOx-control-tech_revised2009152015.pdf

[25] P. R. D'Argenio, G. Barthe, S. Biewer, B. Finkbeiner, and H. Hermanns, "Is your software on dope? - Formal analysis of surreptitiously "enhanced" programs," in *26th European Symposium on Programming, ESOP 2017*, ser. LNCS, vol. 10201. Springer, 2017, pp. 83–110.

[26] S. Biewer, P. R. D'Argenio, and H. Hermanns, "Doping tests for cyber-physical systems," 2018, under submission.

[27] M. A. Köhl, H. Hermanns, and S. Biewer, "Efficient monitoring of real driving emissions," in *Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 11-13, 2018, Proceedings*. Springer, 2018, (forthcoming).

[28] F. Domke, "Personal communication," 2018.

[29] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *POST 2014*, ser. LNCS, M. Abadi and S. Kremer, Eds., vol. 8414.

Springer, 2014, pp. 265–284.

[30] B. Finkbeiner, M. N. Rabe, and C. Sánchez, "Algorithms for model checking HyperLTL and HyperCTL*," in *CAV 2015*, ser. LNCS, D. Kroening and C. S. Pasareanu, Eds., vol. 9206. Springer, 2015, pp. 30–48.

[31] J. Tretmans, "Conformance testing with labelled transition systems: Implementation relations and test generation," *Computer Networks and ISDN Systems*, vol. 29, no. 1, pp. 49–79, 1996.

[32] ——, "Model based testing with labelled transition systems," in *Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers*, ser. Lecture Notes in Computer Science, R. M. Hierons, J. P. Bowen, and M. Harman, Eds., vol. 4949. Springer, 2008, pp. 1–38.

[33] The European Parliament and the Council of the European Union. (2017, June) Commission Regulation (EU) 2017/1151. [Online]. Available: http://data.europa.eu/eli/reg/2017/1151/oj

[34] AVL M.O.V.E Real Driving Emission Testing. Accessed: 2018-07-06. [Online]. Available: https://www.avl.com/emission-measurement/-/asset_publisher/gYjUpY19vEA8/content/avl-m-o-v-e-real-driving-emission-testing

[35] B. D'Angelo, S. Sankaranarayanan, C. Sánchez, W. Robinson, B. Finkbeiner, H. B. Sipma, S. Mehrotra, and Z. Manna, "Lola: Runtime monitoring of synchronous systems," in *12th International Symposium on Temporal Representation and Reasoning (TIME'05)*. IEEE Computer Society Press, June 2005, pp. 166–174.

[36] P. Faymonville, B. Finkbeiner, S. Schirmer, and H. Torfah, "A stream-based specification language for network monitoring," in *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, ser. Lecture Notes in Computer Science, Y. Falcone and C. Sánchez, Eds., vol. 10012. Springer, 2016, pp. 152–168.

[37] The European Parliament and the Council of the European Union, "Directive 98/69/ec of the european parliament and of the council," *Official Journal of the European Communities*, 1998.

[38] G. Traufetter. (2018, May) Audi manipulierte beliebtes Dienstwagenmodell - Produktion gestoppt. [Online]. Available: http://www.spiegel.de/auto/aktuell/audi-manipulierte-beliebtes-dienstwagenmodell-a-1206722.html

[39] The European Parliament and the Council of the European Union. (2007, June) Commission Regulation (EU) 2007/715. [Online]. Available: http://data.europa.eu/eli/reg/2007/715/oj

[40] Audi. (2015, March) Technische Daten - Audi A7 - 3.0 TDI 200 kW s-tronic quattro EU6W.

[41] The International Council on Clean Transportation. (2017) Real-driving emissions test procedure for exhaust gas pollutant emissions of cars and light commercial vehicles in europe. [Online]. Available: https://www.theicct.org/sites/default/files/publications/EU-RDE_policy-update_18012017_vF.pdf