Science and Engineering Ethics (2021) 27:72
https://doi.org/10.1007/s11948-021-00350-5

ORIGINAL RESEARCH/SCHOLARSHIP

Correctness and Completeness of Programming Instructions for Traffic Circulation

Daniela Glavaničová[1] · Matteo Pascucci[2]

Received: 12 November 2020 / Accepted: 28 October 2021 / Published online: 22 November 2021
© The Author(s), under exclusive licence to Springer Nature B.V. 2021

Abstract
In the present article we exploit the logical notions of correctness and completeness to provide an analysis of some fundamental problems that can be encountered by a software developer when transforming norms for traffic circulation into programming instructions. Relying on this analysis, we then introduce a question and answer procedure that can be helpful, in case of an accident, to clarify which components of an existing framework should be revised and to what extent software developers can be held responsible.

Keywords Autonomous vehicles · Encoding rules · Framework revision · Question and answer procedure · Responsibility

Introduction

The possible large-scale release of autonomous vehicles (hereafter, AVs) in the coming years is currently the object of an intense debate between different communities. One of the main issues is understanding how our existing normative systems and infrastructures for traffic circulation have to change in order to accommodate AVs (see, for instance, Douma and Palodichuk 2012). This change will be hopefully reached at the end of a gradual procedure, in which several tests will serve the purpose of detecting and correcting defects of existent frameworks, and will involve vehicles with an increasing level of automation. For instance, in the United States, the *National Highway Traffic Safety Administration* currently (June

✉ Matteo Pascucci
matteopascucci.academia@gmail.com

Daniela Glavaničová
daniela.glavanicova@gmail.com

1 Department of Logic and Methodology of Sciences, Faculty of Arts, Comenius University in Bratislava, Gondova 2, 811 02 Bratislava, Slovak Republic

2 Department of Analytic Philosophy, Institute of Philosophy, Slovak Academy of Sciences, Klemensova 19, 813 64 Bratislava, Slovak Republic

2021) distinguishes between *five progressive levels* of driving assistance technology (as well as a zero level that represents standard vehicles with no automation): this taxonomy describes the foreseen "road to full automation".[1] At Level 1, vehicles are designed with driving assistance features that affect one function at a time. At Level 2, more than one function can be simultaneously controlled by a driving assistance system. At Level 3, an automated system can perform all driving tasks under particular circumstances; however, the human driver is expected to take control if needed, and thus to *constantly* monitor the environment. At Level 4, there are certain circumstances under which the vehicle monitors the environment completely on its own, although in others it may require the human driver to intervene. Finally, at Level 5, the vehicle is able to perform all relevant driving functions on its own under all circumstances, so the human driver is never asked to take control.

Our main focus here will be on the *fifth level* of driving assistance technology, which corresponds to *full automation*; the other four levels involve partial automation only. We will call vehicles of Level 5 *fully autonomous vehicles* (hereafter, FAVs) and vehicles of Levels 1-4 *partially autonomous vehicles* (hereafter, PAVs). Standard vehicles (hereafter, SVs) correspond to vehicles of Level 0. It is sometimes pointed out that FAVs are the long-term objective of the car industry, while PAVs, equipped with devices that have an increasing impact on driving functions, will be released in the initial development stages in order to test the results achieved and to correct possible defects. For this reason, our discussion of FAVs will include occasional references to PAVs as well.

Much attention has been paid to the ethical decisions that an autonomous vehicle is expected to take, as well as to whether these decisions have to be in accordance with some underlying ethical theory and whether they have to be the same for all cars. A detailed survey is offered in Nyholm (2018a) and Nyholm (2018c), as well as in Coeckelbergh (2016). However, here we will reflect on an even more fundamental issue: how can one detect possible defects in an existing framework for traffic circulation and revise it in order to gradually make room for an increasingly automated circulation? Understanding where the flaws lie is also the first step towards fairly attributing responsibility to the various parties involved. Our inquiry will focus on some general problems that, in principle, affect any proposal to transform norms for traffic circulation into programming instructions.

We will base our analysis on the following *three core components*, which can be located at a very high level of abstraction: (i) the set of norms that a machine is expected to observe, (ii) the output of a machine's procedure to acquire norms (i.e., the norms that the machine actually learns), and (iii) the set of actions that a machine actually performs. We will refer to these three components as: the *expected norms*, the *learned norms* and the *performed actions*. A mismatch between these three components can be causally relevant to an accident; when it is properly detected, it can be used as evidence to correct defects of the framework and to clarify responsibility issues.

[1] https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety#topic-road-self-driving.

It is important to stress from the beginning that software developers have to encode *two different levels of norms*, one representing *traffic conventions* and the other instantiating *ethical principles*. Traffic conventions are described in a highway code and depend on the country, type of road, etc. Ethical principles protect various parties from various perspectives. To balance these perspectives, we will follow the intriguing idea of a *complex moral assessment* suggested by Epting (2019). As indicated above, however, we will not enter into the domain of ethics. Our arguments are not based on any particular ethical theory; we rather aim at contributing to the development of *general guidelines* for revising an existing framework (or for developing a new framework), which can be based on any ethical theory whatsoever.

The article is arranged as follows. In "Properties of an Encoding and Priorities Among Rules", we will discuss norms governing traffic circulation and problems associated with their encoding. Our analysis will stress the importance of issues concerning the *correctness* and *completeness* of the encoding, two notions that are inherited from the context of formal logic and that, in our opinion, can be fruitfully exploited in this area. In fact, they witness that defects of an encoding can be of a different kind. In "A Question and Answer Procedure for Investigations", we will describe a question and answer decision procedure that shows how the conceptual framework developed can be put to work in investigations into undesired events that occur within traffic circulation. Our procedure is a preliminary tool aimed at clarifying which components of an existing framework for traffic circulation failed to work in the case of an accident, and to what extent any of the parties involved—in particular, the software developers—can be held responsible. The latter issue will be the focus of "Insights Into Responsibility Attribution".

Properties of an Encoding and Priorities Among Rules

The main task allocated to a software developer consists in transforming the set of norms that a vehicle is expected to observe into a set of rules learnt by the machine and governing its behaviour. In our opinion, it is important to highlight two preliminary points about this. The first point is that there is always a human behind an AV. Indeed, despite their name, AVs behave on the basis of how they are designed and programmed, as well as on the basis of the infrastructure in which they are supposed to circulate. This is the case both for machines programmed via deductive methods and for those programmed via non-deductive methods since, in any case, the set of norms that a machine learns depends on the way in which the learning procedure is designed by humans. Thus, when we examine what happens to AVs on the road, we ultimately have to go back to some human decisions and how these decisions are implemented.

The second point is that AVs have to deal with uncertain scenarios. In this regard, the software of an AV has to compensate for a human's ability to foresee possibly dangerous scenarios during circulation, an aspect that is frequently mentioned in traffic regulations and not easily translatable (if at all) into a set of rules. For instance, when an unexpected obstacle suddenly appears, the machine has to follow strategies for trajectory replanning that aim at simulating what we would expect a human driver to do in those circumstances (Subosits and Gerdes 2019). Moreover, sensors used by an AV to gather information about the environment have limited capabilities, and the AV's decisions need to take into account uncertainty about what could happen in the immediate future (Thornton 2018). Of course, this is a delicate issue and the decision taken by a machine cannot be expected, *per se*, to always ensure an optimal result. As a matter of fact, in similar scenarios we would not even expect humans to make choices able to ensure optimal outcomes. This also means that state uncertainty leading to a decision has to be taken into account in order to reasonably attribute responsibility in cases of accidents.

Now, taking the two points made above for granted, we will take a closer look at the basic features of an encoding, by which we mean any procedure of transforming the set of norms that a machine is expected to observe during circulation into the set of norms that a machine actually learns and that, subsequently, guide the machine's behaviour. Thus, our analysis of an encoding relies on the relation among the three core components mentioned in Sect. 1: expected norms, learned norms and performed actions.

Norms for AVs are usually specified by some agency which is authorized by the state where cars circulate, such as the *National Highway Traffic Safety Administration* (2013) in the US. We can imagine that these norms can be theoretically grouped according to the set of circumstances to which they are supposed to apply. For instance, let us assume that N is a consistent set of norms and that it is intended to apply to a set of circumstances C during traffic circulation. An encoding of N is a set of instructions $\mathtt{I}$ which should aim to satisfy two fundamental properties of correctness and completeness, when given as input to a machine α that is not affected by malfunctioning or external interference (e.g., by hackers). We provide rigorous definitions of these two properties below, accompanied by some comments. The notion of instruction employed in the definitions is assumed to have a broad meaning, encompassing both deductive and non-deductive rules. On the basis of this assumption, we will hereafter use the terms 'instruction' and 'learned norm' as interchangeable.

Definition 1 *(Correctness of the encoded norms)* A set of instructions $\mathtt{I}$ governing the behaviour of a machine α correctly encodes a set of norms N applicable to a set of circumstances C if and only if there is no $\mathsf{c} \in \mathsf{C}$ in which compliance with $\mathtt{I}$ leads α to violate N.

Thus, correctness means here that all learned norms are also expected norms. Correctness with respect to a set of norms N is a very general property. In fact, it may be satisfied by different sets of instructions, each allowing for a certain range of possible actions of a machine. In particular, two sets of instructions $\mathtt{I}_1$ and $\mathtt{I}_2$ may both be correct with respect to a set of norms N, even though the range of possible actions of a machine under $\mathtt{I}_1$ is larger than under $\mathtt{I}_2$. The following simple scenario (formulated in terms of single instructions and of a single norm) exemplifies this point. Let α and β be two cars circulating on a road r; moreover, let i_1 and i_2 be instructions given to α and to β, respectively. While i_1 allows α to travel at a speed up to 100 km/h, i_2 allows β to travel at a speed up to 80 km/h: α has an additional range of speed options if compared with β. Despite this difference, compliance with the received instruction ensures that neither α nor β will ever violate a norm stating that the speed limit on r is 120 km/h.

Definition 2 *(Completeness of the encoded norms)* A set of instructions I governing the behaviour of a machine α completely encodes a set of norms N applicable to a set of circumstances C if and only if, for every c ∈ C, I enables α to comply with N in c.

Thus, completeness means here that learned norms are sufficient for the machine to get a representation of all expected norms in all relevant circumstances. Also in this case, it is worth observing that completeness with respect to a set of norms N may be satisfied by different sets of instructions, each allowing for a certain range of possible actions of the machine. However, all these alternative sets will share a common core of instructions that are followed by the machine whenever it complies with N. The difference between two complete sets of instructions will rather rely on the representation of norms that are not in N. Most importantly, this entails that, if N is the set of all norms applicable to traffic circulation, then two complete sets of instructions with respect to N will diverge on aspects that are not fundamental to circulation (e.g., specific settings for passenger comfort).

Ensuring the completeness of a set of instructions is certainly a hard challenge for software developers, especially when the set of norms N to be encoded is very large. Arguably, it is a problem that can never be solved in its full generality, due to the extreme variety of possible scenarios during traffic circulation. Yet, further reflection on the set C of the relevant circumstances to which a set of norms N applies could provide some hints towards a way of addressing this problem that would be at least satisfactory. In fact, a machine α is often authorized to find itself only in *some* of the relevant circumstances to which a set of norms N applies. This is traditionally the case in frameworks for traffic circulation. For instance, some standard vehicles cannot circulate on (or have limited access to) controlled-access highways; therefore, even if N is a set of norms applying to all sorts of highways, such vehicles have to comply with N only in a restricted subset of all the relevant circumstances. The future circulation of various categories of autonomous vehicles could be likewise differentiated, depending on the way in which each category is designed; accordingly, the general problem of formulating (correct and) complete instructions to encode a set of norms could be decomposed into specific problems for each category of vehicles. Moreover, in order to deal with completeness issues in computational models of traffic circulation during a test phase, one could start with the analysis of a small subset of all the relevant circumstances for a given set of norms and, subsequently, expand this set in a gradual way, with the analysis of new circumstances. For instance, at the beginning one could focus on one category of vehicles, one category of streets and a limited set of possibly interacting parties. Checking whether completeness could be achieved with a set of instructions in such context would still be difficult, but comparatively much easier than in contexts with unrestricted options. Then, once a sufficient level of approximation to the desired outcome would be reached, one could stepwise add other categories of vehicles/streets/interacting parties and check the result.

However, there is also another aspect of the problem to deal with. In fact, Definitions 1 and 2, taken on their own, do not take into account the internal structure of sets of norms. Let us assume that the specified set N corresponds to the set of all norms mentioned in a certain legal text regulating traffic circulation, such as a highway code. A correct and complete encoding of all the norms of a highway code in the sense specified above is not enough to ensure a safe circulation of AVs in highways. Indeed, while two norms having the same surface structure (e.g., an obligation) will be codified in the same way, the importance of their content might differ. Thus, some norms contained in a highway code need to take priority over others when they are transformed into instructions. This is not only the case between two norms n and n' such that n explicitly represents an exception to n' (or a reparation for the violation of n'), but also between two norms n and n' that are not immediately interrelated. For the mentioned reasons, an encoding should also aim to satisfy the following two properties (for the sake of simplicity, we assume a one-one correspondence between expected norms and learned norms (i.e., instructions governing a machine's behaviour; from a logical perspective, this correspondence can easily be obtained by replacing a set of instructions with a unique instruction representing their conjunction):

Definition 3 *(Correctness of the encoded hierarchy)* For no pair of instructions i_1 and i_2, such that i_1 encodes a norm n_1 and i_2 encodes a norm n_2, the following holds:

- i_1 overrides i_2 while n_1 does not override n_2.

Correctness here means that the hierarchy among learned norms does not add any priority relation to the hierarchy of expected norms.

Definition 4 *(Completeness of the encoded hierarchy)* For no pair of instructions i_1 and i_2, such that i_1 encodes a norm n_1 and i_2 encodes a norm n_2, the following holds:

- i_1 does not override i_2 while n_1 overrides n_2.

Completeness here means that the hierarchy among expected norms is fully reproduced by the hierarchy of learned norms.

Another problem is due to norms that apply to specific traffic circumstances but are not directly expressed in a highway code. For instance, norms concerning the safety conditions of the various parties involved in traffic circulation, such as those prescribing the protection of pedestrians' lives. Drivers of SVs are normally aware that similar norms are in effect, due to their background knowledge, and they also know that these norms are more important than others that are explicitly mentioned in the highway code.

Let us consider, as an example of some of the issues discussed, a scenario taking place on a road in Italy. A pedestrian decides to cross the road at a point where there is no zebra-crossing and a driver of an SV is approaching the pedestrian. The pedestrian starts crossing the road without giving precedence to the car. Shortly after, the driver detects the pedestrian and realizes that the only way to avoid an accident would be to temporarily move to the emergency lane. Article 141(2) of the Italian Highway Code says that a driver has to always maintain control of her vehicle and be able to perform all necessary actions in safety conditions, especially halting the vehicle within the limits of her visibility and in front of any obstacle that can be foreseen.[2] From this generic norm, one can infer the specific norm that drivers have to avoid hitting pedestrians crossing the road, even if the latter happen to behave in a way that is forbidden. In our scenario the pedestrian is actually behaving in a forbidden way, since crossing a road outside the zebra-crossing and without giving precedence to cars represents a violation of Article 190(5) of the Italian Highway Code. Finally, Article 3(15) of the Code says that drivers of SVs are not allowed to use the emergency lane, except in the case of emergency stops (and other articles of the Code say that an emergency stop can take place either when the driver is no longer able to drive, due to a sudden physical impairment, or when the vehicle undergoes some sudden technical malfunction). If the driver knows that the norm prescribing the safeguarding of pedestrians' lives is more important than the norm that forbids travelling in the emergency lane (as we would expect), she can take the right decision to prevent an accident.

By contrast, how can we expect an FAV to take the right decision in an analogous situation? A program can compensate for the lack of any human driver's ability to intervene only via an *appropriate ordering* among the learned norms. In deductive programming approaches this may mean assigning a different weight to different commands. In non-deductive programming approaches this may mean associating additional rewards to the machine's desired behaviour while formulating the decision-making algorithm. In general, one has to distinguish, at least, between *two levels of norms*: those that represent *conventions of traffic circulation* (such as not travelling in the emergency lane) and those that represent *fundamental ethical principles* (such as safeguarding pedestrians' lives). We think that rules of the second level should always take priority over rules of the first level when there is a conflict, since, as the example discussed above shows, this can make a difference in terms of protecting the lives of certain parties involved in traffic circulation.[3] We also want to stress that the suggested priority is completely independent of any underlying ethical theory. Clearly, such a priority issue should be clarified already at the level of norms

[2] The text of The Italian Highway Code (Codice della Strada) is available online at https://www.normattiva.it/uri-res/N2Ls?urn:nir:stato:legge:1992;495!vig=.

[3] Of course, there are very important ethical principles that cannot be expected to take priority in the case of a conflict with traffic conventions, and hence cannot be included in the class of fundamental principles *in this particular context*. For instance, the principle of minimizing emissions of pollutant gas is extremely important for environmental issues, but it can never force one to contravene traffic conventions. It should rather be observed *as far as possible in addition to all traffic conventions*.

that we expect a machine to comply with (*expected norms*), hence before a machine learns the principles actually guiding its behaviour (*learned norms*).

Ethical principles, both fundamental and non-fundamental ones for traffic circulation, are usually expressed as very generic norms in highway codes (if at all), and for this reason it is harder to transform them into programming instructions. In order to address this problem, a software developer should be provided with *rules (i.e., expected norms) that instantiate ethical principles in specific circumstances*. Consider the principle of safeguarding a pedestrian's life: in the example just discussed it would be instantiated by a specific norm prescribing use of the emergency lane if that is the only way of avoiding a pedestrian detected on the road. This also means that norms expressing ethical principles will be codified via *a set of circumstance-based instructions* corresponding to the set of circumstance-based norms. It is fair to assume that software developers will only have to deal with the final output of the theoretical analysis aimed at converting ethical principles into circumstance-based expected norms. Therefore, any mistake in this procedure will be beyond their range of responsibility.

Conflicts among norms instantiating ethical principles have to be settled by establishing a priority among these principles. Understanding the context-dependent priority of ethical principles is certainly beyond the set of tasks that can be allocated to a software developer. For instance, there might be rules concerning the protection of human life, rules concerning the protection of private property, rules concerning nature preservation, rules concerning the preservation of cultural artifacts, and so forth. Following Epting, we hold that an approach of *complex moral assessment* can help to identify all sorts of ethical principles and how they can be prioritized (Epting 2019, p. 396):

> Implementing AVs in today's cities could affect vulnerable people, and they will also affect the public, nonhumans, future generations, and historically and culturally significant artifacts. Each of these groups deserves consideration for different reasons, but they do not equally require it.

We endorse the core insight of the above passage, that implementing AVs requires establishing priority among various rules rooted in different ethical considerations. Another view we share with Epting is that *the order of priorities is not absolute*. One source of this relativity is that cities differ. Implementing AVs in an ancient city would differ radically from implementing AVs in a modern city, or in a village near a nature reserve. For instance, imagine that the Parthenon is to be demolished in order to facilitate the transportation of a certain group of vulnerable people. In that context, we might want to set our priorities differently. Notice, however, that this relativity does not affect *what takes priority within a certain category* (e.g., which buildings are rated as more important and which are rated as less important); it rather affects *which categories take priority*. In other words, it can be justifiable to prioritize one category over another in a certain context, despite the fact that the latter category is usually prioritized.

A Question and Answer Procedure for Investigations

The idea of two levels of rules for AVs and the specification of a class of fundamental ethical principles can be exploited to establish a first fundamental priority relation among learned norms. Furthermore, a complex moral assessment in the sense of Epting (2019) helps to refine the hierarchy by adding context-dependent priorities among ethical principles. The result of this procedure can be used to provide some guidelines for assessing and revising an existing framework; moreover, it provides ground for attributing responsibility when something goes wrong in the circulation of AVs. Since our focus is on FAVs, we will not deal with issues related to the liability of drivers who are able to take temporary control of cars. In this section we will describe a question and answer procedure which is intended to help establish the mentioned guidelines. Clearly, our procedure is not intended to determine once and for all all possible sources of flaws in the framework or who is responsible for what; it rather provides a strategy to examine events and leaves open the possibility that further evidence is needed to make corrections and reach normative conclusions. Thus, it can be regarded as a starting point for building a rigorous decision procedure for investigations based on the theoretical framework that we have laid down in the previous sections.

Before describing the procedure, we stress a methodological point. In order to look for potential causes of undesired events involving AVs, one can proceed by relying either on direct experience of real-life scenarios or on computational simulations of real-life scenarios. In both cases, anyway, one acquires experimental evidence that can be used to assess the functioning of the overall framework designed for the circulation of AVs. Thus, a procedure like the one described below can be used for corroborating a model of traffic circulation either before or during its real-life implementation.

The procedure starts with input information about something that went wrong in (a simulation of) a scenario S and chooses a particular machine, which we can call α_1, to examine. The first question employed in the procedure targets the core problem of norm encoding: whether α_1 was able to act in accordance with the learned norms applicable to S. In other words, this amounts to asking whether performed actions give rise to any compliance issues with respect to learned norms. We will label this question as (Q1), the affirmative answer as (A1+) and the negative answer as (A1−). For this question we will first analyse the affirmative answer, whereas we will normally proceed in the opposite order for other questions.

(A1+) *The car* α_1 *was able to act in accordance with all learned norms (applicable to* S). This means that there is something wrong with the set of norms that α_1 learned or that there was external interference. The information available is not yet sufficient to draw any conclusion on flaws of the framework that have to be corrected or to make any liability judgement.

Without loss of generality, the procedure can be arranged in such a way that it first checks any problems due to the set of learned norms, and later any problems due to external interference. Accordingly, the next objective is to find reasons for the mentioned problems. However, since learned norms should encode expected norms, at this stage it is convenient to go back to the root of the theoretical framework and check whether there are already issues in the set of expected norms. Therefore, the second question employed by the procedure is whether the norms α_1 was expected to follow in S are consistent (Q2).

(A2−) *The set of expected norms is not consistent.* In this case, there is a mistake made by people to whom the task of specifying norms for AVs was allocated. For instance, the problems might boil down to a decision taken either by a committee of experts on ethical principles, or by a committee of experts on traffic conventions. A subset of the relevant norms has to be revised to eliminate the discovered inconsistency. One can look at the available evidence in order to determine to what extent the experts' mistake could have been prevented, and whether any liability proposal can be made at this stage.

(A2+) *The set of expected norms is consistent.* In this case, there are two possibilities: (i) if we assume that the set of expected norms regulating the circulation of AVs in a certain context is sufficient to cover all possible scenarios, as we currently do in the case of norms regulating the circulation of SVs (e.g., the Italian Highway Code, plus the set of ethical principles referred therein, is assumed to be sufficient to cover all possible scenarios arising in Italian roads), then experts on ethical principles or on traffic conventions can be ruled out as possible bearers of responsibility, and the problems encountered are more likely due to the encoding of the norms. By contrast, (ii) if we get rid of such an assumption, the mentioned experts can still be held responsible for undesired events. This second possibility may arise in cases of moral uncertainty due to original features brought into the framework by AVs, an aspect that is further addressed below in answer (A4−b) of the procedure.

Since encoding norms is a task that includes several components, we need to look at these components in order to determine to what extent software developers can be held responsible for what happened. This is a crucial part of investigations aimed at detecting possible flaws in an existing framework, since different flaws may lead to different outputs of the procedure. In other words: what we want to stress is that not all defects of a framework are of the same kind. More specifically, we will proceed by distinguishing between encoding problems related to correctness and encoding problems related to completeness. We will argue that, in some cases, correctness and completeness issues make a difference with respect to responsibility. These notions can be used both with reference to sets of norms (see Definitions 1 and 2) and with reference to hierarchies (see Definitions 3 and 4). We then proceed by discussing hierarchies, since problems with this component can again lead investigations back to the root of the framework (as when sets of norms are inconsistent). Thus, the next question employed by the procedure is whether the encoding of the hierarchy of norms produced the desired results (Q3), namely, whether its output matches the hierarchy among expected norms. We first consider the negative answer, which amounts to a mistake made by software developers, and then the positive answer, which opens the possibility of questioning the original hierarchy of norms.

(A3−) *The hierarchy of learned norms does not match the hierarchy of expected norms.* There are two sub-cases, depending on whether the issue is due to correctness (a) or completeness (b).

(A3−a) *The hierarchy of expected norms is not correctly encoded in instructions.* We think that in this case there are ingredients to potentially consider software developers responsible for what happened (yet, these ingredients might still be insufficient for a proper responsibility ascription). Indeed, consider two instructions i_1 and i_2, which encode two norms n_1 and n_2, respectively. A decision that i_1 has to take priority over i_2, while the relevant normative text does not indicate a corresponding priority of n_1 over n_2, looks unmotivated.

(A3−b) *The hierarchy of expected norms is not completely encoded in instructions.* At a first glance, one might be tempted to apply the same criteria used in (A3−a) and conclude that software developers are likely liable for what happened, since they failed to recognize a priority relation among the norms (see Definition 4). However, sometimes such relations are not explicit in the text in which the norms are described (e.g., a highway code or, even more significantly, a list of ethical principles). Hence liability is a more delicate issue in this case, and there might be some ambiguity in the original source of the norms: this needs to be taken into account in any overall evaluation of the encoding.

(A3+) *The hierarchy of norms is encoded as expected.* We then need to look for further criteria in order to understand what caused the problem encountered in our scenario S.

At this point, we consider the two levels of norms that have to be encoded and proceed by analysing the higher level, the one involving ethical principles. Therefore, the fourth question we ask is whether the encoding of norms instantiating ethical principles produced the expected results (Q4).

(A4−) *The learned norms on fundamental ethical principles do not match the expected ones.* Again, we need to analyse both correctness (a) and completeness (b) issues.

(A4−a) *The expected norms on fundamental ethical principles are not encoded correctly.* This means that there is at least one instruction provided to the machine which directly conflicts with a norm that instantiates an ethical principle. On the one hand, we would be inclined to say that software developers are somehow liable for what happened. On the other hand, encoding ethical principles correctly is not an easy task. Indeed, as we have said, these principles have to be converted into circumstance-specific norms; and the latter, in turn, have to be converted into a set of circumstance-specific instructions. Such a conversion procedure is error-prone, in the sense that the original meaning of a generic norm n concerning ethical principles might be lost along the way. In our opinion, a software developer can be held liable for a mismatch between a circumstance-specific norm (the penultimate element of the conversion) and its codifying instruction (the last element), but not for a mismatch between a generic norm and the set of its codifying instructions.

(A4−b) *The expected norms on fundamental ethical principles are not encoded completely.* Since instantiating norms for fundamental ethical principles *exhaustively* is even more difficult than specifying some correct circumstance-specific instance, we think that when the provided instructions do not completely cover all possible circumstances, programmers are generally not liable to sanctions. This view is endorsed by Goodall (2014, p. 99), while discussing the implementation of ethical theories via programming instructions: "Rules can be added or clarified to cover different situations, but it is unclear if any set of rules could encompass all situations."[4] In the case of moral uncertainty about a particular kind of scenario, an even more fundamental problem emerges: there might be a set of circumstances that remain, from the beginning, normatively unsettled (see, e.g., the discussion by Bhargava and Kim 2017).

(A4+) *The learned norms on fundamental ethical principles match the expected ones.* Ruling out problems due to the encoding of norms concerning ethical principles, the next step is to look at the encoding of norms of the first level.

The fifth question we ask is thus whether conventions of traffic circulation are encoded in an appropriate way (Q5).

(A5−) *The learned norms on conventions of traffic circulation do not match the expected ones.* This is either a problem of correctness (a) or a problem of completeness (b).

(A5−a) *The expected norms on conventions of traffic circulation are not encoded correctly.* This means that there is at least one instruction provided to the machine which directly conflicts with a traffic convention. Here software developers seem to be liable for the mistake, since conventions of traffic circulation are clearly specified in the adopted highway code, and their encoding is far less puzzling than the encoding of norms concerning ethical principles. However, as usual, one should carefully consider the specific mistake made.

(A5−b) *The expected norms on conventions of traffic circulation are not encoded completely.* We can basically apply the same criteria used in (A5−a). The idea is that, ultimately, any software unable to encode at least conventions on traffic circulation in a faithful and exhaustive way has to be regarded as a non-excusable flaw of the framework.

(A5+) *The learned norms on conventions of traffic circulation match the expected ones.* The investigation needs to proceed further.

At this point it seems that the reason for what happened is that the car was unable to follow all the instructions, namely that there was a mismatch between learned norms and performed actions. Therefore, we need to go back to the first question and analyse what follows from a negative answer.

(A1−) *The car α_1 was unable to act in accordance with all learned norms (applicable to* S). We need to look at other candidates for attributing responsibility, since the software developers, apparently, fulfilled all their duties.

There might have been some problem due to α_1's hardware, or other AVs involved in the scenario, or the infrastructure in which the AVs were circulating, or some

[4] In addition to our comments on Definitions 1 and 2, a. possible way of addressing this issue is, in our opinion, thinking of frameworks for AVs as analogous to frameworks designed for the circulation of some automated means of public transportation. For instance, consider the case of automated trains: they have a limited capability of spatial motion due to the presence of tracks and, for this reason, their behaviour can be more easily foreseen. Similarly, early frameworks for AVs could allow these vehicles to perform a limited amount of spatial moves and to have limited forms of interactions among them, all of which would be easy to foresee and well regimented by the available software. This would also require keeping AVs separated from other vehicles and human agents (cyclists, pedestrians, etc.) in such early stages, in order to avoid any unforeseen event due to the other categories of road users.

unforeseen event in which AVs played no active role. Each of these cases gives rise to new questions that complete the procedure. We will analyse this part of the investigation more concisely, since our aim was to focus on the duties of software developers.

We first ask whether α_1 was designed appropriately (Q6). A negative answer (A6−) means that the manufacturers made some mistake, so they could be liable to some sanction. A positive answer (A6+) leads one to the next stage of the inquiry, that is, what was the behaviour of the other AVs involved in the scenario? At this point one has to ask questions (Q1)-(Q6) for each of the other AVs. These questions have to be iterated finitely many times, since the number of cars involved in the scenario is finite. If in every case one ends by reaching the answer (A6+), then the next step is to ask whether the infrastructure for traffic circulation was appropriate (Q7). A negative answer to the latter question (A7−) means that the civil engineers who created the infrastructure for traffic circulation did something wrong, so they might be held accountable. A positive answer to the latter question (A7+) rules out the liability of the civil engineers. The last option is to check whether any unforeseen event interfered with the behaviour of the AVs involved (e.g., a bolt of lightning struck a car, or a pedestrian crossed the road in a thoughtless way). This corresponds to the final question (Q8). In the case of a positive answer (A8+), all the categories of people involved in setting up the circulation of AVs are not liable to sanctions, and there might be some non-AV party involved in the accident who bears responsibility (e.g., a pedestrian or an SV). In the case of a negative answer (A8−), the event remains unexplained on the basis of the current normative system, so the normative system should be revised to take similar scenarios into account.

Insights Into Responsibility Attribution

Various possible bearers of responsibility have been suggested as a solution to the problems around responsibility in the context of autonomous vehicles. The present focus is, however, rather narrow: responsibility of a software developer. Given this, the aim of this section is two-fold: (i) to clarify what kinds of responsibility the software developers can bear; and (ii) to use the proposed question and answer procedure to shed some light on cases when the software developers (should) bear responsibility.

To begin with, a software developer can bear historic responsibility for something that has happened or a future-oriented prospective responsibility (on this distinction, see Cane 2002, Ch. 2). They can be responsible for bad things, as well as for neutral or even good things. Furthermore, they can be responsible in the *legal*, *moral*, or *causal* sense.

Considering the above options, let us note that our prime focus will be on prospective responsibility understood as task allocation, and on historic responsibility understood as the ascription of a causal contribution. In our opinion, these notions are a good starting point for evaluating a framework for automated circulation, for two reasons. First, delineating how duties are distributed among the various parties involved in preparing the framework is fundamental to understanding which component has to be improved if anything fails to work. Second, causal responsibility can provide a basis for attributing moral responsibility or legal liability. As White and Baum (2017, p. 66) observe, "in general, the law punishes those who have caused harm, particularly the harm that could and should have been avoided."

As far as moral and legal responsibility are concerned, we will briefly discuss which notions of such responsibility of the software developers enter the picture. Software developers can be understood as individual or collective entities. If the former treatment is assumed, there are the following options, which are due to the general understanding of individual responsibility: an individual can be morally or legally responsible for an intentional wrongdoing, for negligent or reckless wrong, and can be strictly responsible.[5]

For instance, a software developer can intentionally encode the hierarchy of norms incorrectly. To give a more concrete example, imagine a scenario from Lin (2014): "An autonomous car is facing an imminent crash. It could select one of two targets to swerve into: either a motorcyclist who is wearing a helmet or a motorcyclist who is not. What's the right way to program the car?" Now imagine that the hierarchy of norms indicates a preference to safeguarding parties that comply with the norms (hence, the motorcyclist wearing a helmet). Imagine further that the software developer in question has a partner motorcyclist who intentionally never wears a helmet. They reflect this in the encoding and thereby encode the hierarchy incorrectly, shifting the priority. This would be a case of intentional wrongdoing.

Alternatively, it might happen that a software developer is not sure how to read the hierarchy of norms they are encoding: should it be understood in a way that a motorcyclist who is wearing a helmet or a motorcyclist who is not wearing it should be preferred in the case of an accident? Let us assume they could, in principle, consult an expert who would surely help to settle the matter, but they do not: the developer, well aware of the risk, decides to proceed the way they think is more likely. This can be understood as an individual recklessness.

Another alternative still is that a software developer for some reason does not maintain the required standard of care (for instance, an insufficient amount of checking is done). As a consequence, they fail to encode the hierarchy of norms in a correct way. This can be understood as a case of negligence.

Or imagine that there was no intentional wrongdoing, no recklessness, no negligence: a software developer did their best. But still some norms are not correctly encoded, and this has fatal consequences. This can be understood as a case of strict responsibility; on strict responsibility (or rather, strict answerability), moral and criminal, see Duff (2009).

If there is not a single software developer responsible for a clearly defined part of the encoding, but a whole group of them working together, collective responsibility might be a good alternative to individual responsibility.[6]

[5] We note here already that there is no consensus on how to define these notions precisely; nevertheless, our points will be neutral in the way that they will carry over to various proposals in the literature.

[6] Collective responsibility then can be understood as boiling down to individual responsibility, or as irreducible collective. For the former approach, see Narveson (2002); for the latter, see Isaacs (2011).

To give an example, imagine that one software developer takes care of norms regulating preferences in the context of a car crash with respect to the vulnerability of those involved; another one takes care of all norms regulating the behaviour of an AV with respect to the presence of individuals (not) wearing protective equipment, such as helmets; a group of others take care of norms regulating the presence of motorcyclists in the traffic, and so forth. The encoding of the norms under discussion can thus be a result of a joint work of several individuals. In that case, one can still assign responsibility to the individuals for their contributions, but only to some extent. When it comes to all undesired outcomes that cannot be attributed to any individuals, a space opens for collective responsibility (or alternatively, for responsibility gaps).

Final Remarks

In this paper we have employed two notions taken from formal logic – correctness and completeness – to detect (and group) mistakes in an existing framework for traffic circulation. We have developed a theoretical framework that is based on the the notions of ‘expected norms’ (that is, the norms that we expect AVs to comply with), ‘learned norms’ (that is, the norms that AVs actually learn via any programming approach, either deductive or non-deductive), and ‘performed actions’ (that is the actions actually performed by AVs). Our analysis has addressed possible mismatches between these three components which are shared by different frameworks. Naturally, detecting and grouping mistakes in an existing framework for traffic circulation is a promising starting point for improving this framework. This is the first advantage of the proposed theoretical work.

The second advantage is that our analysis can shed some light also on the responsibility ascription in the context of AVs, albeit in a limited extent only. Throughout the paper, we have focused primarily on the role played by a software developer, and in turn, on responsibility that might attach to this party. We have clarified that a software developer might be treated as an individual or as a collective entity. In both cases, we have sketched the kinds of responsibility that can be attributed to software developers, ranging from intentional wrongdoing, recklessness, or negligence to strict responsibility.

Acknowledgements Daniela Glavaničová was supported by the Slovak Research and Development Agency under the contract no. APVV-170057 and VEGA 1/0197/20. Matteo Pascucci was supported by the Štefan Schwarz Fund for the project “A fine-grained analysis of Hohfeldian concepts” (2020-2022) and by the VEGA Grant No. 2/0117/19. The authors thank their colleagues at the Department of Analytic Philosophy of the Slovak Academy of Sciences for useful comments on this work.

Author Contributions The contents of the article are the result of a joint research work of the two authors.

References

Bhargava, V., & Kim, T.W. (2017). Autonomous vehicles and moral uncertainty. In: P. Lin, K. Abney, & R. Jenkins (Eds.), *Robot ethics 2.0. From autonomous cars to artificial intelligence* (pp. 5–19). Oxford University Press.

Cane, P. (2002). *Responsibility in law and morality*. Hart Publishing.

Coeckelbergh, M. (2016). Responsibility and the moral phenomenonology of using self-driving cars. *Applied Artificial Intelligence, 30*(8), 748–757.

Douma, F., & Palodichuk, S. A. (2012). Criminal liability issues created by autonomous vehicles. *Santa Clara Law Review, 52*(4), 1157–1169.

Duff, A. (2009). Legal and moral responsibility. *Philosophy Compass, 4*(6), 978–986.

Eliot, L. (2019). Key to driverless cars, Operational Design Domains (ODD), here's what they are, woes too. *Medium*, April 19.

Epting, S. (2019). Automated vehicles and transportation justice. *Philosophy & Technology, 32*(3), 389–403.

Friedrich, B. (2016). The effect of autonomous vehicles on traffic. In M. Maurer, J. C. Gerdes, B. Lenz, & H. Winner (Eds.), *Autonomous driving* (pp. 317–334). Springer.

Goodall, N.J. (2014). Machine ethics and automated vehicles. In: G. Meyer, & S. Beiker (Eds.), *Road vehicle automation* (pp. 93–102). Springer.

Grunwald, A. (2016). Societal risk constellations for autonomous driving. Analysis, historical context and assessment. In M. Maurer, J. C. Gerdes, B. Lenz, & H. Winner (Eds.), *Autonomous driving* (pp. 641–663). Springer.

Gurney, J. (2017). Applying a reasonable driver standard to accidents caused by autonomous vehicles. In: P. Lin, K. Abney, & R. Jenkins (Eds.), *Robot ethics 2.0* (pp. 51–65). Oxford University Press.

Hevelke, A., & Nida-Rümelin, J. (2015). Responsibility for crashes of autonomous vehicles: An ethical analysis. *Science and Engineering Ethics, 21*(3), 619–630.

Isaacs, T. (2011). *Moral responsibility in collective contexts*. Oxford University Press.

Lin, P. (2013). The ethics of saving lives with autonomous cars are far murkier than you think. *Wired*. July 30.

Lin, P. (2014). The robot car of tomorrow may just be programmed to hit you. *Wired*. May 6.

Lin, P. (2016). Why ethics matters for autonomous cars. In M. Maurer, J. C. Gerdes, B. Lenz, & H. Winner (Eds.), *Autonomous driving* (pp. 69–85). Springer.

Loh, W., & Loh, J. (2017). Autonomy and responsibility in hybrid systems. In: P. Lin, K. Abney, & R. Jenkins (Eds.), *Robot ethics 2.0* (pp. 35–50). Oxford University Press.

Marchant, G. E., & Lindor, R. A. (2012). The coming collision between autonomous vehicles and the liability system. *Santa Clara Law Review, 52*(4), 1321–1340.

McFarland, M. (2015). Google's chief of self-driving cars downplays 'the trolley problem'. *The Washington Post*, December 1.

Narveson, J. (2002). Collective responsibility. *The Journal of Ethics, 6*(2), 179–198.

National Highway Traffic Safety Administration (2013). Preliminary Statement of Policy concerning Automated Vehicles. 12–14 May.

Nyholm, S. (2018a). The ethics of crashes with self-driving cars: a roadmap, I. *Philosophy Compass*, e12507.

Nyholm, S. (2018). The ethics of crashes with self-driving cars: a roadmap. II. *Philosophy Compass*, e12506

Nyholm, S. (2018). Attributing agency to automated systems: Reflections on human-robot collaborations and responsibility loci. *Science and Engineering Ethics, 24,* 1201–1219.

Siddiqui, F. (2019). What self-driving cars can't recognize may be a matter of life and death. *The Washington Post*, November 11.

Subosits, J. K., & Gerdes, J. C. (2019). From the racetrack to the road: Real-time trajectory replanning for autonomous driving. *IEEE Transactions on Intelligent Vehicles, 4*(2), 309–320.

Taylor, M. (2016). Self-driving Mercedes-Benzes will prioritize occupant safety over pedestrians. *Car and Driver*, October 7.

Thornton, S.M. (2018). Autonomous vehicle speed control for safe navigation of occluded pedestrian crosswalk. Available at: arxiv:1802.06314

White, T.N., & Baum, S.D. (2017). Liability for present and future robotics technology. In: Lin, P., Abney, K., & Jenkins R. (eds.), *Robot ethics 2.0* (pp. 66–79). Oxford University Press.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.