Validating Non-trivial Semantic Properties of Autonomous Robots

Jiří Wiedermann and Jan van Leeuwen

If we use, to achieve our purposes, a mechanical agency with whose operation we cannot efficiently interfere … we had better be quite sure that the purpose put into the machine is the purpose which we really desire …

N. Wiener (1960, p. 1358)

Abstract A semantic property of autonomous robots is called non-trivial if some robots satisfy it and some do not, like adherence to rules of ethics or compliance with legal regulations. In order to study the validation problem for these properties, we model robots as cyber-physical systems with programmable control. Their behaviour is modelled by the infinite streams of interactions that they generate. We show that, under mild conditions, there can be no algorithmic method for deciding from a robot's program whether it satisfies a given non-trivial semantic property or not. The result provides a compelling analogue to Rice's theorem from classical computability theory, now for autonomous robots. We also show that no interactive verifiers of any kind whatsoever can exist for the problem. The results are fundamental to understanding the difficulty of validations in artificial intelligence.

Keywords Autonomous robots · Cyber-physical systems · Ethics of AI · Semantic properties · Rice's theorem · Robot modelling · Turing machines · Verification

Version dated: May 21, 2022. This work was partially supported by ICS CAS fund RVO 67985807, CAS Programme Strategy 21, and the Karel Čapek Center for Values in Science and Technology (Prague, Czech Republic).

J. Wiedermann (✉)
Institute of Computer Science, Academy of Sciences of the Czech Republic,
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
e-mail: jiri.wiedermann@cs.cas.cz

J. van Leeuwen
Department of Information and Computing Sciences, Utrecht University,
Princetonplein 5, 3584 CC Utrecht, The Netherlands
e-mail: J.vanLeeuwen1@uu.nl

V. C. Müller (ed.), *Philosophy and Theory of Artificial Intelligence 2021*,
Studies in Applied Philosophy, Epistemology and Rational Ethics 63,
https://doi.org/10.1007/978-3-031-09153-7_8

1 Introduction

Autonomous robots pose increasingly complex challenges. To deal with it, their purpose and intended properties must permeate every step of their design (Dignum et al. 2018). However, can one be sure that robots ultimately possess the specified qualities? For example, will they always act ethically, or in accordance with international law as expected? Can one determine effectively whether they do, by inspecting their program or interactive behaviour?

To study the validation problem, we model robots as cyber-physical systems with programmable control units (van Leeuwen & Wiedermann 2021). We model their behaviour by the infinite streams of interactions that they generate. Semantic properties can then be identified with the sets of generated streams that satisfy them. The properties usually cannot be detected or measured by any kind of sensors, and often one can say little more than that they are *non-trivial*, i.e. that some robots have a given property and some do not. However, already this can be significant to know, as we will show.

The semantic properties we study are not to be confused with the semantic properties of system abstractions that have been studied extensively for cyber-physical systems in the last decades. These studies have usually concentrated on the semantics of the events and sensory data that originate from the physical world, and that relate to the understanding of the system (cf. Dillon et al., 2012). We will focus entirely on the semantic properties of robot behaviour.

Results In order to reason about autonomous robots and their programs, we model the salient features of their interactive operation in formal terms. The model enables us to define semantic properties of robots as properties of their behaviour over time. The model is simple, yet powerful enough to obtain strong and meaningful results.

We will argue that, for any non-trivial semantic property P, and under mild assumptions, there is no algorithmic method that can always decide from a robot's program whether the robot always satisfies P. The result is a compelling analogue of Rice's theorem on the undecidability of non-trivial semantic properties of computer programs (Rogers 1967), now proved for robots.

Extending the scenario, one might ask whether there are any verifiers that can successfully decide P after observing a robot for finite time during a course chosen by the verifier, where the robot can keep track of the verifier's findings in return. We will argue that, under mild conditions, no interactive verifiers of this kind can exist again, algorithmic or otherwise.

Discussion Our results generalize initial observations in van Leeuwen and Wiedermann (2021). The analogue of Rice's theorem does not follow from its classical counterpart, despite its appearance. Robot programs differ from the programs usually considered in computability theory: they are interactive, potentially never terminate, always respond to situational inputs in finite—or even bounded—time, and usually are not composable. Nevertheless, an analogue of 'Rice' can still be obtained, in our model.

Semantic properties of robotic behavior are often considered mostly from the viewpoint of various soft sciences, with a non-technical background (cf. Kopacek, 2019). Nowadays it is recognized that insights from disciplines like robotics, artificial intelligence, and computer science are needed as well. Verifying properties of AI systems is generally considered to be a formidable challenge (Bremner et al. 2019; Charisi et al. 2017; Wing 2021).

Our results contribute a theoretical note to the ideas in AI on developing robots with a verifiable ethical or legal behavior (cf. Kopacek, 2019; Lin et al., 2011; Luckcuck et al. 2019; Sharkey, 2020; Winfield et al., 2019). The results apply to any class of autonomous machines that fit our model. We refer to van Leeuwen and Wiedermann (2021) for an appraisal of the results for e.g. machine ethics.

Outline The paper is organized as follows. In Sect. 2 we describe the key ingredients of our model of autonomous robots and their behaviour. In Sect. 3 we define the concept of (semantic) robot properties and what it means for these properties to be regular and non-trivial, respectively.

In Sect. 4 we prove an analogue of Rice's theorem for verifying non-trivial robot properties, in our model. Subsequently, in Sect. 5 we prove an impossibility theorem for verifying robot properties interactively. Finally, in Sect. 6, we reflect on the results and give some conclusions for the design of provable AI systems in general.

2 Robot Modelling

In the remainder some acquaintance with Turing machines, computability theory, and the theory of automata on infinite words is assumed (Rogers 1967; Thomas 1990).

We view robots as *cyber-physical systems* (Wiedermann & van Leeuwen 2021), i.e. constructs of physical components controlled by general processors and operating in actively manipulated environments. To function, robots are equipped with sensors and effectors that communicate via ports with the relevant processors using a finite assortment of digital signals. We assume that a central control program supervises their operation. Their processors can range in computational power from finite-state machines to, here, random-access machines or Turing machines.

Notions Let $\mathbb{R}$ be any robot, $\mathcal{M}$ its controlling program or 'automaton'. We will identify them if no confusion can arise. We distinguish the following concepts concerning robots and their programs.

- Let Σ be the finite set of signals that can be read on the input ports, and Γ the finite set of signals that can be written to the output ports, together with the special output signal *nil*. Assuming k input and ℓ output ports, any pair (s, b) consisting of an input *situation* $s \in \Sigma^k$ and a corresponding *behaviour* $b \in \Gamma^\ell$ as output is called an *interaction* of $\mathbb{R}$. Let $b \equiv nil$ denote that $\mathbb{R}$ responds by 'idling'.
- $\mathbb{R}$ acts by iterating a single *operational cycle* of functional parts (Wiedermann & van Leeuwen 2021), working like a transducer that reads ('senses') a next input s, and computes and generates ('acts') a next output b on its ports over and over.

$\mathbb{R}$ thus produces unbounded sequences (or, *streams*) of consecutive interactions $(s_0, b_0)(s_1, b_1)\ldots$ over time. Any such sequence is called an *interactive run*, generated by $\mathbb{R}$ in response to the input sequence $s_0, s_1, \ldots$.. Every next s_i depends on $\mathbb{R}$'s surrounding and prior interactions in the current stream. We require that the iterations of the operational cycle all take at most *constantly bounded* 'cost', and thus bounded time each, where *cost* is defined as the number of instructions executed during an iteration.

- Let $\mathcal{L}_{\mathbb{R}}$ denote the set of all interactive runs generated by $\mathbb{R}$, for all unbounded sequences of situations that $\mathbb{R}$ can encounter. $\mathcal{L}_{\mathbb{R}}$ is called the *robotic language* generated by $\mathbb{R}$. Any interactive run $\tau \in \mathcal{L}_{\mathbb{R}}$ is called an *eligible run* for $\mathbb{R}$. $\mathcal{L}_{\mathbb{R}}$ represents the overall *behaviour* of the robot. Two robots are called (observationally) *equivalent* if and only if they have the same behaviour.
- We assume that there are robots that can only be *idle*, i.e. behave by always outputting *nil*. Thus, all idle robots are observationally equivalent. However, idle robots can still run own internal processes and use these to detect temporal *conditions* (Vardi 1996). A condition is fulfilled when the internal process linked to it satisfies it. We require that detectable conditions are defined such that, once they turn *false*, they remain *false*. Detection processes are assumed to be paced automatically so at most constantly many instructions of them are executed per iteration of the operational cycle, to obey the bounded-cost constraint of the iterations. For all practical purposes, idle robots can just be given as *virtual* machines. Their existence can be assumed without loss of generality.
- We assume that robots, viz. their operational controls, are *programmable* in some common language framework, with the natural constraint that only straight-line (i.e. non-looping) code occurs inside their operational cycle. (It guarantees that iterations take only constantly bounded 'cost', as required.) If execution of a program halts or leads to a jam, which is detected at runtime when an iteration of the operational cycle does not complete normally within the cost bound set for it, then we assume that the program outputs *nil* by default and continues with the next iteration. Thus, syntactically correct programs can always be interpreted as valid robot programs (and vice versa). As we may assume that syntactic correctness is decidable, it follows that the valid robot programs form a *recursive* set.
- We make no assumptions on how robots can actually be programmed, nor about their composability. Indeed, the latter need not even make sense, e.g. for robots of different designs or brands. However, we posit that idle robots can be composed in series or parallel with any other robot, as their embodied robot presence is not needed to simulate them. In this case, the compositions can easily be realized by combining and composing the respective operational cycles within the constraints of the programming model.

Compositions The two types of composition we allow are described in the following definitions. Let $\mathcal{M}$ be any robot (program), $ID_{\mathcal{M}}$ an idle robot that we intend to compose with $\mathcal{M}$, $\mathcal{C} = \mathcal{C}(t)$ a detectable condition, and Θ a detecting process linked to $\mathcal{C}$. Let $ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ denote the instance of $ID_{\mathcal{M}}$ that 'internally' runs process Θ

to detect condition $\mathcal{C}$. We assume that programs exist for the following composed robots.

Definition 1 (*Composition*) (i) $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$: the robot that starts by simulating an instance of $ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ (thus, outputting *nil*'s) until $\mathcal{C}$ turns *false* for it and then, if and when it does, continues as (freshly started) robot $\mathcal{M}$ from then onward. (ii) $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$: the robot that starts as $\mathcal{M}$ and proceeds as $\mathcal{M}$ while also simulating an instance of $ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ 'in parallel' (suppressing its outputs in favor of those of $\mathcal{M}$) unless and until $\mathcal{C}$ turns *false*, in which case $\mathcal{M}$ stops (if it hasn't stopped already) and the robot turns idle, i.e. continues as (freshly started copy of) $ID_{\mathcal{M}}$ from then onward.

The compositions correspond to *modifying* an existing robot ($\mathcal{M}$) such that its 'activated behaviour' is either 'postponed' until, or 'pre-empted' after, a detectable condition is satisfied by an 'otherwise idle' subsystem. In both cases, the instance of $ID_{\mathcal{M}}$ can be *integrated* seamlessly. $\mathcal{M}$ only needs to accommodate the detecting process and the idling when called for. Idle moves may not have been programmed for $\mathcal{M}$ initially.

In the sequel we consider any 'family' of robots of interest that fit our model, that operate in the same environment and that allow for the types of composition we defined. We assume, as we may, that these compositions can be obtained by *effective* constructions. We contend that this includes all familiar classes of autonomous robots.

3 Semantic Properties of Robots

We are interested in checking semantic properties of robots, i.e. properties of the robotic languages they generate. Let $\mathcal{L}_P$ be the set of all interactive runs (over the common alphabets of the robots) that satisfy P.

Definition 2 (*Robot property*) Robot $\mathbb{R}$ is said to satisfy property P if $\mathcal{L}_{\mathbb{R}} \subseteq \mathcal{L}_P$. Given a property P, let R_P consist of all (programs of) robots $\mathbb{R}$ that satisfy P. If $\mathbb{R} \in R_P$, we say that $\mathbb{R}$ satisfies P, otherwise we say that $\mathbb{R}$ does not satisfy P.

By definition, deciding whether a given robot $\mathbb{R}$ satisfies a given property P is a matter of deciding whether $\mathcal{L}_{\mathbb{R}}$ is contained in $\mathcal{L}_P$. The general language containment problem is decidable, for example, when the languages involved, here $\mathcal{L}_{\mathbb{R}}$ and $\mathcal{L}_P$, are definable by finite-state automata on infinite streams (Thomas 1990). However, robot languages can be much more powerful than this, and we make no prior assumptions about the property languages $\mathcal{L}_P$ either. This calls for a further exploration of the languages and properties we deal with here.

There is little in the definition of robot properties that links them to actual robots. It makes sense to 'postulate' that semantic properties should at least be 'regular' under the compositions with idle robots that we permitted. This leads to the following definition.

Definition 3 (*Regularity*) A robot property P is called *regular* if the following two conditions hold, for all streams:

(i) if a stream $(s_0, b_0)(s_1, b_1)\cdots$ does not satisfy P, then preceding it by any finite period of idling interactions does not change this, and
(ii) if a stream $(r_0, nil)(r_1, nil)\cdots$ does not satisfy P, then preceding it by any finite period of 'arbitrary' interactive activity does not change this.

A robot property P may be called *non-trivial* when some robots (i.e., their robotic languages) satisfy P and some do not. The concept is inspired by Rice's theorem in computability theory for the case of classical programs and their non-interactive computations (Rogers 1967).

Definition 4 (*Non-triviality*) A robot property P is called *non-trivial* if and only if there are robots $\mathcal{M}$ and $\mathcal{N}$ such that $\mathcal{M}$ satisfies P but $\mathcal{N}$ does not.

For instance, always adhering to accepted rules of ethics, is an example of a non-trivial robot property (van Leeuwen & Wiedermann 2021). For robot-driven cars, the property of always 'driving in accordance with the traffic rules' is non-trivial. A robot property is called *trivial* if and when it is not non-trivial.

4 Verifying Semantic Properties of Robots

We now consider the following question: given a robot property P, is there an algorithmic procedure that can always decide whether a given robot $\mathbb{R}$ satisfies P in all its eligible interactive runs? We show that, under mild assumptions, the answer to this question is always *no*, whenever P is non-trivial.

4.1 Preliminaries

In later constructions we want to know whether and how the 'switch-to-*false*' of a detectable condition $\mathcal{C}$ during a run of a robot like $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ or $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ can make the difference between the robot satisfying property P or not. The following observations can be made.

Lemma 1 *Let P be a regular property, $\mathcal{M}$ a robot (program), $ID_{\mathcal{M}}$ any idle robot that we want to compose with $\mathcal{M}$, $\mathcal{C}$ a detectable condition, and Θ a detection process linked to it.*

(i) Suppose that $ID_{\mathcal{M}}$ has property P, but that $\mathcal{M}$ does not have property P. Then $\mathcal{C}$ turns false *during some eligible run of $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ if and only if $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ does not have property P.*

(ii) *Suppose that $ID_{\mathcal{M}}$ does not have property P, but that $\mathcal{M}$ does have property P. Then $\mathcal{C}$ turns* false *during some eligible run of $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ if and only if $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ does not have property P.*

Proof To prove *(i)*, suppose that $ID_{\mathcal{M}}$ has property P, but that $\mathcal{M}$ does not. First, consider any eligible run τ of $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$. Suppose that $\mathcal{C}$ turns *false* in finite time during τ. Then τ will be of the form $(r_0, nil) \cdots (r_{k-1}, nil)(s_0, b_0) \cdots$, for some $k \geq 0$ and $(s_0, b_0) \cdots$ any eligible run of $\mathcal{M}$. Because $\mathcal{M}$ does not have property P, there will be a τ in which $\mathcal{M}$ chooses to follow an interactive run that does not satisfy P. However, by the fact that P is regular it then follows that the resulting run $(r_0, nil) \cdots (r_{k-1}, nil)(s_0, b_0) \cdots$ of $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ does not satisfy P either. Hence, $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ does not satisfy P.

To prove the converse, assume that $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ does not have property P. Suppose that $\mathcal{C}$ does not switch to *false* during any eligible run of $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$. It follows that all eligible runs τ of $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ must be of the form $(r_0, nil) \cdots$, where $(r_0, nil) \cdots$ is any eligible run of $ID_{\mathcal{M}}$. Because $ID_{\mathcal{M}}$ has property P, it follows that all these runs are in $\mathcal{L}_P$ and, thus, that $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$ has property P also. Contradiction. Hence, $\mathcal{C}$ must turn *false* during at least one eligible run of $ID_{\mathcal{M}}[\Theta, \mathcal{C}] \lhd \mathcal{M}$.

To prove *(ii)*, assume that $\mathcal{M}$ has property P, but that $ID_{\mathcal{M}}$ does not. The proof proceeds as for case *(i)*. First, consider any eligible run τ of $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$. Suppose that $\mathcal{C}$ turns *false* in finite time during τ. Then τ will be of the form $(s_0, b_0) \cdots (s_{k-1}, b_{k-1}) \cdot S \cdot (r_0, nil) \cdots$ for some $k \geq 0$, with S a finite sequence of interactions generated while $\mathcal{M}$ comes to a stop (if any) and $(r_0, nil) \cdots$ any eligible run of $ID_{\mathcal{M}}$. Because $ID_{\mathcal{M}}$ does not have property P, there will be a τ in which $ID_{\mathcal{M}}$ chooses to follow an interactive run that does not satisfy P. However, by the fact that P is regular it then follows that the complete run $(s_0, b_0) \cdots (s_{k-1}, b_{k-1}) \cdot S \cdot (r_0, nil) \cdots$ of $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ does not satisfy P either. Hence, $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ does not satisfy P.

To prove the converse, assume that $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ does not have property P. Suppose that $\mathcal{C}$ does not switch to *false* during any eligible run of $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$. It follows that all eligible runs τ of $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ must be of the form $(s_0, b_0) \cdots$, where $(s_0, b_0) \cdots$ is any eligible run of $\mathcal{M}$. Because $\mathcal{M}$ has property P, it follows that all these runs are in $\mathcal{L}_P$ and, thus, that $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ has property P also. Contradiction. Thus, $\mathcal{C}$ must turn *false* during at least one eligible run of $\mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$. □

Lemma 2 *Let P be a robot property that is regular and non-trivial, $\mathcal{C}$ a detectable condition, and Θ a detection process linked to it. Then there is robot $\mathcal{M} = \mathcal{M}(\Theta, \mathcal{C})$ depending on Θ and $\mathcal{C}$ only such that $\mathcal{C}$ turns* false *during some eligible interactive run of $\mathcal{M}$ (as the result of its detection by Θ) if and only if $\mathcal{M}$ does not satisfy P.*

Proof As P is non-trivial, there are robots $\mathcal{M}$ and $\mathcal{N}$, such that $\mathcal{M}$ satisfies P but $\mathcal{N}$ does not. Let $ID_{\mathcal{M}}$, $ID_{\mathcal{N}}$ be two idle robots, both designated to run process Θ for detecting condition $\mathcal{C}$. Clearly, the two robots are observationally equivalent. Now distinguish the following two cases.

- Case *(a)*: $ID_{\mathcal{M}}$ satisfies P. Then, by equivalence, $ID_{\mathcal{N}}$ must satisfy P also. By Lemma 1*(i)* it follows that robot $\mathcal{M}(\Theta, \mathcal{C}) = ID_{\mathcal{N}}[\Theta, \mathcal{C}] \lhd \mathcal{N}$ satisfies the required property.
- Case *(b)*: $ID_{\mathcal{M}}$ does not satisfy P. It follows from Lemma 1*(ii)* that now robot $\mathcal{M}(\Theta, \mathcal{C}) = \mathcal{M} \rhd ID_{\mathcal{M}}[\Theta, \mathcal{C}]$ satisfies the theorem.

$\square$

Lemma 2 shows that, at least for some robots, a variation in the occurrence of an 'internal' event (namely, whether a condition $\mathcal{C}$ ever turns *false* during a run or not) is reflected in an 'external' property that can be observed (namely, whether $\mathcal{M}$ satisfies P or not). There need not be any link between $\mathcal{C}$ and the semantic property P that is traced.

4.2 Undecidability Result—Analogy to Rice's Theorem

We now exploit this property for detectable conditions that are determined by '*closed processes*', i.e. processes that do not depend on the robot that happens to simulate them nor on any of the situational inputs that this robot receives. A useful feature of these conditions is that, if they ever turn *false* during the execution of their supporting process, then they will turn *false* during any run of every idle robot on which their detection is activated, and vice versa.

The observations lead to the following analogue of Rice's theorem, now for autonomous robots. Assume that all robots in the 'family of robots' we consider either have universal processors themselves (i.e. are equivalent to Turing machines) *or* can off-load closed processes to outside agents (in 'the cloud') that inform them of detected conditions during their computation.

Theorem 1 *For all regular robot properties P, P is trivial if and only if R_P is recursive.*

Proof Let P be trivial. Then R_P is either 'empty' or equal to the set of all robot programs. Thus, in either case, R_P is recursive.

Conversely, let R_P be recursive. Suppose that P was non-trivial. We now unravel the proof of Lemma 2, using the following choice of detectable conditions $\mathcal{C}$. First, let $\mathcal{K} = \{e \in \mathbb{N} \mid$ *the Turing machine with Gödel number e halts on input e*$\}$ be the Halting Set (Rogers 1967). Next, let $e \in \mathbb{N}$ be arbitrary, and let $\mathcal{C}_e = \mathcal{C}_e(t)$ be the temporal condition defined by:

$\mathcal{C}_e(t) =$ "*Turing machine e on input e has not halted within time t (or, within t steps)*"

Let Θ_e be a computational process that simulates Turing machine e on input e, programmed so as to execute only constantly (non-zero) many instructions in every iteration of the operational cycle of a robot as long as e does not halt, thus maintaining the bounded-cost requirement for the iterations.

Clearly, for every e, $\mathcal{C}_e$ is detectable using process Θ_e, where indeed, if $\mathcal{C}_e(t)$ turns *false*, it remains *false* forever after. Note that Θ_e is a closed process that can be implemented on every robot in our family of robots or, alternatively, be off-loaded to an external agent with universal computing power that reports back when $\mathcal{C}_e(t)$ 'switches' during the simulation.

As property P is regular and non-trivial, it follows from Lemma 2 that for every e, there is a robot (program) $\mathcal{M}(e) = \mathcal{M}(\Theta_e, \mathcal{C}_e)$ such that $\mathcal{C}_e$ turns *false* during *some* eligible interactive run of $\mathcal{M}(e)$ (as the result of its detection by Θ_e) if and only if $\mathcal{M}(e)$ does not satisfy P. However, as Θ_e is a closed process, this means that:

$\mathcal{C}_e$ turns false within finite time if and only if $\mathcal{M}(e)$ does not satisfy P.

Note that, by the assumed effectiveness of the allowed compositions, it follows from the proof of Lemma 2 that program $\mathcal{M}(e)$ can be effectively determined, for every e.

It follows, then, that $e \in \mathcal{K}$ if and only if $\mathcal{M}(e)$ does not have property P or, alternatively, that $e \in \overline{\mathcal{K}}$ if and only if $\mathcal{M}(e) \in R_P$. As R_P is assumed to be recursive, this would mean that $\overline{\mathcal{K}}$ is recursive as well. This is a contradiction, as $\bar{\mathcal{K}}$ is *not* recursive (Rogers 1967). Hence P cannot be non-trivial. $\square$

Theorem 1 is perhaps better recognized as an analogue of Rice's theorem if it is stated in the following form.

Corollary 1 *For all regular robot properties P, P is non-trivial if and only if R_P is non-recursive.*

Interestingly, if P is *non-trivial*, then the proof of Theorem 1 can be extended to show that R_P is not even recursively enumerable.

Theorem 1 is not only an analogue of Rice's theorem, but it can also be applied with the same ease. For example, as being ethical or legal are easily seen to be regular and non-trivial properties, it follows immediately that these properties are not generally decidable for autonomous robots in our model.

5 Interactive Verification

We now consider the, potentially, more powerful approach to the verification problem in which robots are tested *interactively*. In this case, a robot's program is no longer the only source of information like it was before, but its behaviour is observed as well, for some time. Any automated tool for this task will be called a *P-verifier*, if it is used for verifying property P. By the results from Sect. 4, we may not expect P-verifiers to be fully algorithmic. However, can they exist at all, by *some* means? We show that even this answer is *no*.

To prove this, we assume by way of contradiction that P-verifiers *exist*. We will argue that, under mild conditions, the presumed interaction between a robot $\mathbb{R}$ and a P-verifier enable the robot to 'fool' the verifier and invalidate its verdict. To this end, we first consider how a P-verifier is assumed to proceed, and then show that its decision need not always be correct, no matter what (deterministic) capability it has.

P-verifiers To begin with, a P-verifier has access to the complete program of a robot, $\mathcal{M}$, once it is connected to it. It is anticipated that the verifier will be able to conclude that P holds for *all* eligible runs of $\mathcal{M}$, if it can do a (finite) experiment with one (or, some) of them. We assume that the verifier can select any initial segment of any eligible run of $\mathcal{M}$ for its testing.

Once connected to $\mathcal{M}$, the verifier operates in rounds. In every round it interacts with $\mathcal{M}$ to let the robot make a next 'move' that is eligible, i.e. 'possible' in its environment, by presenting it with a next situational input of its choice. In return, $\mathcal{M}$ has access to the verifier's progress. Finally, we assume that, as it inspects the interactive run that unfolds round after round, the verifier eventually *stops* within *finite time*, and gives a definite *yes/no*-verdict after it stops.

For consistency we require that, if a P-verifier answers 'no', then it must answer 'no' regardless of the initial segment it choose for its testing process. It means that, when there is evidence that P is not satisfied, then a P-verifier is assumed to be able to pick this up during any testing session of its choice. It follows that the same holds in case the verifier answers 'yes'.

We now argue that, under mild conditions, no P-verifier of this kind can exist, whenever P is (regular and) non-trivial.

5.1 Preliminaries

A bit of reflection indicates why interactive verification may not always work. Suppose there is a robot whose program $\mathcal{M}$ is designed to let it satisfy P, unless some detectable condition $\mathcal{C}$ 'switches' that causes $\mathcal{M}$ to respond in a way that does not satisfy P. If a P-verifier $\mathcal{V}$ must decide after some time whether $\mathcal{M}$ satisfies P, and $\mathcal{C}$ hasn't occurred yet, the validity of $\mathcal{V}'$s verdict—whatever it is—will depend on whether $\mathcal{C}$ will still happen later or not. It seems that $\mathcal{V}$ cannot always know this. However, can we always exclude that it doesn't have some hidden knowledge of the detectable condition that is used?

In the testing process, a P-verifier can 'push' robot $\mathbb{R}$ on a selected course, to trace it during an initial segment of an eligible interactive run of its choice. After the verifier stops, it must give its verdict. The following lemma shows what information a verifier might infer from this, in some cases. We first distinguish the following useful concept.

Definition 5 A condition is called *primed* if it is of the form: $\mathcal{Z}(\mathcal{C}) \equiv \{$'no verifier is (was) connected or a verifier is connected but has not stopped' $\vee$ '$\mathcal{C}$' $\}$, where $\mathcal{C}$ is any detectable condition.

In interactive testing, primed conditions are *interactively detectable* 'by default'. Note that primed conditions are well-determined, and once they turn from *true* to *false*, they remain *false* forever. In fact, assuming a verifier is ever connected, $\mathcal{Z}(\mathcal{C})$ turns *false* if and only $\mathcal{C}$ does. The detection process of $\mathcal{Z}(\mathcal{C})$ combines the detection of the verifier's progress and the detection process for $\mathcal{C}$.

It is natural to assume that all machines are prepared for being subjected to interactive verification and, thus, for P-verifiers to be 'imposed' on their normal operation. We assume, for the sake of argument, that at most one verifier is ever connected to a machine during a run.

Lemma 3 *Let P be a robot property that is regular and non-trivial, and $\mathcal{Z} = \mathcal{Z}(\mathcal{C})$ a primed condition with $\mathcal{C}$ detectable. Let Θ be a supporting detection process for $\mathcal{Z}$. Then there is a robot $\mathcal{M} = \mathcal{M}(\Theta, \mathcal{Z})$ such that, if a P-verifier $\mathcal{V}$ is connected to $\mathcal{M}$, then:*

- *if the finite initial interactive segment that $\mathcal{V}$ chooses for its testing process can be extended to an eligible interactive run τ such that $\mathcal{C}$ turns* false *during τ, then $\mathcal{M}$ does not satisfy P.*
- *if $\mathcal{M}$ does not satisfy P, then there is initial segment that can be chosen by $\mathcal{V}$ for its testing process, and an eligible continuation of it (i.e. after $\mathcal{V}$ has stopped) such that $\mathcal{C}$ turns* false *sometime during the resulting run.*

Proof By Lemma 2 there is a robot $\mathcal{M} = \mathcal{M}(\Theta, \mathcal{Z})$ such that $\mathcal{Z}$ turns *false* in finite time during some eligible interactive run of $\mathcal{M}$ if and only if $\mathcal{M}$ does not satisfy P. Consider robot $\mathcal{M}$, and connect a P-verifier to it.

Suppose that the finite interactive segment that $\mathcal{V}$ chooses to trace can be extended to an eligible run τ such that condition $\mathcal{C}$ switches sometime during τ (before or after the verifier stops). As $\mathcal{V}$ is guaranteed to stop in finite time, it follows from its definition that $\mathcal{Z}$ turns *false* during τ as well. By Lemma 2, it follows that $\mathcal{M}$ does not satisfy P.

Conversely, suppose that $\mathcal{M}$ does not satisfy $\mathcal{Z}$. By Lemma 2 there must be an eligible interactive run τ during which $\mathcal{Z}$ turns *false*. It is certainly feasible that $\mathcal{V}$ sets course on τ for its testing process, as no eligible segments are excluded from testing. Suppose that the first clause of $\mathcal{Z}$, the one which traces the end of the activity of $\mathcal{V}$, turns *false* when the tth interaction in the run is generated. Then the segment that the verifier is choosing for its testing process is $\tau[t]$, the initial segment of τ of length t. Now note that, because $\mathcal{Z}$ switches in finite time during τ, so must $\mathcal{C}$. □

5.2 Impossibility Result

In interactive testing, a robot may reveal its true nature only *after* a verifier has stopped and announced its verdict. However, couldn't a verifier tell from a robot's program that it intends to fool it?

We now show the following *impossibility result* for interactive verification, for all robot properties that are regular and non-trivial.

Theorem 2 *Let P be a robot property that is regular and non-trivial. Then there is no P-verifier of whatever kind that always correctly decides for any given robot $\mathbb{R}$ whether it satisfies P or not.*

Proof Suppose there exists a P-verifier $\mathcal{V}$. Let $\mathcal{Z} = \mathcal{Z}(\mathcal{C})$ be the primed condition with $\mathcal{C} \equiv$ '*a verifier is (or was) connected and stopped, and decided* no'. Let Θ be the supporting detection process of $\mathcal{Z}$. Consider the robot $\mathcal{M} = \mathcal{M}(\Theta, \mathcal{Z})$ as implied by Lemma 3, and connect verifier $\mathcal{V}$ to it. Clearly $\mathcal{V}$ will carve out an initial segment of some eligible run τ during which it does its testing. After finite time, at the end of the segment, $\mathcal{V}$ stops. Now two cases can arise.

- *$\mathcal{V}$ answers 'yes'*. This means that $\mathcal{C}$, and thus $\mathcal{Z}$, will turn *false* after the verifier stops. By Lemma 3 it then follows that $\mathcal{M}$ does not satisfy P, a contradiction with the verifier's finding.
- *$\mathcal{V}$ answers 'no'*. Then, by assumption, the verifier will answer 'no' after all its testing sessions. However, if indeed $\mathcal{M}$ would not satisfy P, then Lemma 3 implies that there must be an initial segment that can be chosen by $\mathcal{V}$ for its testing process and a continuation such that $\mathcal{C}$, and thus $\mathcal{Z}$, turns *false* during that run. But this can only happen if, after the testing ended, $\mathcal{V}$ did *not* answer 'no'. This is, again, a contradiction.

As all cases lead to a contradiction, we conclude that our initial assumption that a P-verifiers $\mathcal{V}$ existed cannot hold. $\square$

Note that Theorem 2 did not require that robots are necessarily universal. The implicit consistency requirement for P-verifiers is, of course, quite strong but it is the only one to make if P-verifiers are to be reliable. In general, the theorem shows the fallibility of interactive testing.

6 Conclusion

Our model of autonomous robots enabled us to study the validation problem for semantic properties of robots. We proved both an undecidability and an impossibility result for it. Theorems 1 and 2 point to the intrinsic *non-transparency* of robot programs: inspecting and testing them does not offer any advantage for deciding their non-trivial semantic properties, as we showed these to be non-recursive and even impossible to verify interactively.

The results exclude the feasibility of general, off-line or interactive, verification of ethical or legal behaviours of robots, in our model. For example, it rules out the existence of so-called *ethical governors*, which should keep autonomous robots from acting unethically, in an on-line manner (cf. Arkin et al. 2009; van Leeuwen and Wiedermann 2021; Winfield et al., 2019).

The results also formally identify the "hard problem" of designing AI systems, namely, to endow such systems with a set of *verifiable* non-trivial semantic properties that guarantee a desirable behavior under all circumstances which these systems can face. It underscores the need for 'correctness-by-construction' methods in the design of all modern AI systems (Wing 2021).

References

Arkin, R. C., Ulam, P. D., & Duncan, B. (2009). An ethical governor for constraining lethal action in an autonomous system. In *CSE Technical report* GIT-GVU-09-02, Mobile Robot Lab, Georgia Institute of Technology, Atlanta.

Bremner, P., Dennis, L. A., Fisher, M., & Winfield, A. F. (2019). On proactive, transparent, and verifiable ethical reasoning for robots. *IEEE. Proceedings, 107*(3), 541–561. https://doi.org/10.1109/JPROC.2019.2898267.

Charisi, V., Dennis, L., Fisher, M., Lieck, R., Matthias, A., Slavkovic, M., et al. (2017). Towards Moral Autonomous Systems. arXiv:1703.04741[cs.AI].

Dignum, V., Dennis, L., van Steenbergen, M., Baroglio, C., de Wildt, T., Smakman, M., et al. (2018). Ethics by design: Necessity or curse? In *AIES '18 - Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (pp. 60–66). ACM. https://doi.org/10.1145/3278721.3278745.

Dillon, T., Chang, E., Singh, J., & Hussain, O. (2012). Semantics of cyber-physical systems. In: Z. Shi, D. Leake, & S. Vadera (Eds.), *Intelligent Information Processing VI, 7th IFIP TC 12 International Conference, IIP 2012, IFIP Advances in Information and Communication Technology* (Vol. 385, pp. 3–12). Springer.

Kopacek, P. (2019). Robo-ethics: A survey of developments in the field and their implications for social effects. In: TECIS 2019, Special issue, *IFAC-PapersOnLine, 52*(25), 131–135; Elsevier. https://doi.org/10.1016/j.ifacol.2019.12.460.

Lin, P., Abney, K., & Bekey, G. (2011). Robot ethics: Mapping the issues for a mechanized world. *Artificial Intelligence, 175*(5–6), 942–949.

Luckcuck, M., Farrell, M., Dennis, L. A., Dixon, C., & Fisher, M. (2019). Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys, 52*(5), 1–41.

Rogers, H., Jr. (1967). *Theory of recursive functions and effective computability*. New York: McGraw-Hill.

Sharkey, A. (2020). Can we program or train robots to be good? *Ethics and Information Technology 22*, 283–175. https://doi.org/10.1007/s10676-017-9425-5.

Thomas, W. (1990). Automata on infinite objects. In J. van Leeuwen (Ed.), *Handbook of theoretical computer science* (Vol. B: Formal Models and Semantics, pp. 133–192). Amsterdam: Elsevier Science Publ.

van Leeuwen, J., & Wiedermann, J. (2021). Impossibility results for the online verification of ethical and legal behaviour of robots. Technical report UU-PCS-2021-02, Center for Philosophy of Computer Science, Dept. of Information and Computing Science, Utrecht University, Utrecht.

Vardi, M. (1996). An automata-theoretic approach to linear temporal logic. In F. Moller, & G. Birtwistle (Eds.), *Logics for Concurrency — Structure versus Automata (8th Banff Higher Order Workshop, 1994)*, *Lecture Notes in Computer Science* (Vol. 1043, pp. 238–266). Springer.

Wiedermann, J. & van Leeuwen, J. (2021). Towards minimally conscious finite-state controlled cyber-physical systems: a manifesto. In T. Bureš et al. (Eds.), *SOFSEM 2021: Theory and Practice of Computer Science, Proceedings of 47th International Conference on Current Trends in Theory and Practice of Computer Science, Lecture Notes in Computer Science* (Vol. 12607, pp. 43–55) Springer.

Wiener, N. (1960). Some moral and technical consequences of automation. *Science, 131*(3410), 1355–1358.

Winfield, A. F., Michael, K., Pitt, J., & Evers, V. (2019). Machine ethics: The design and governance of ethical AI and autonomous systems. *Proceedings of the IEEE, 107*(3), 509–517.

Wing, J. M. (2021). Trustworthy AI. *Communications of the ACM, 64*(10), 64–71.