Autonomics: In search of a foundation for next-generation autonomous systems

David Harel[a,1], Assaf Marron[a], and Joseph Sifakis[b]

[a]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 7610001, Israel; and [b]Verimag Laboratory, Université Grenoble Alpes, 38400 Saint Martin d'Hères, France

This contribution is part of the special series of Inaugural Articles by members of the National Academy of Sciences elected in 2019.

Contributed by David Harel, June 3, 2020 (sent for review February 19, 2020; reviewed by Carl K. Chang, Werner Damm, and Moshe Y. Vardi)

The potential benefits of autonomous systems are obvious. However, there are still major issues to be dealt with before developing such systems becomes a commonplace engineering practice, with accepted and trustworthy deliverables. We argue that a solid, evolving, publicly available, community-controlled foundation for developing next-generation autonomous systems is a must, and term the desired foundation "autonomics." We focus on three main challenges: 1) how to specify autonomous system behavior in the face of unpredictability; 2) how to carry out faithful analysis of system behavior with respect to rich environments that include humans, physical artifacts, and other systems; and 3) how to build such systems by combining executable modeling techniques from software engineering with artificial intelligence and machine learning.

autonomous systems | autonomics | trustworthy systems

Autonomous systems are already able to replace humans in carrying out a variety of functions. This trend will continue in the years to come, with autonomous systems becoming central and crucial to human society. They will be broadly prevalent and will include, e.g., vehicles of all kinds, medical and industrial robots, agricultural and manufacturing facilities, and distributed management for traffic, urban security, and electric grids.

Many organizations are already striving to develop the next wave of trustworthy, cost-effective autonomous systems, and researchers are busy building powerful tools and methods for the development process. However, extremely high levels of complexity and criticality present fundamental new challenges. Consider, for example, even a very modest autonomous system, a valet-parking robot—obviously a far cry from a full autonomous vehicle (AV). Customers and regulators would be fully justified in asking whether the robot will be able to discover a child forgotten in the car, or notice that a pet dog is perched underneath it. Even if it is able to notice these, what will the robot do as a result? How will it react if a human attempts to stop it by pursuing it and yelling?

Next-generation autonomous systems will be expected to operate under conditions that will often be unpredictable at the time of their development, due to limited control over the system's environment, the dynamic emergence of new kinds of objects and events in the world, and the exponential growth in the number of composite configurations of such elements, old and new alike. The literature contains interesting demos and discussions of unpredictability in autonomous systems; see, e.g., the (simple) SpotMini household-assistant video (1) and, more importantly, the discussion/critique of its demonstration environment (2). While test environments and simulation engines provide ever-increasing variation and realism [see, e.g., CARLA (3), PARACOSM (4), HEXAGON MSC (5), and Cognata (6)], these are still constrained and synthetic. Engineers must be able to assure customers and regulators that the system will function correctly and safely, not only in a large variety of critical scenarios, but also in complex high-risk situations never even thought about previously.

There is a growing awareness that the challenges of developing next-generation autonomous systems will be difficult to accomplish, due to weaknesses in established methods and processes. In other words, challenges such as those described above cannot be dealt with by merely enhancing the system's safety features, say, by adding sensors, actuators, and logic, and/or by carrying out richer test cases. Indeed, in order to address some of these challenges, the combined community that consists of relevant groups in industry, government, and academia, is launching roadmapping activities and large-scale collaborative projects (7–10).

Still, we argue, this is not enough. The required trustworthiness mandates different, and more fundamental, advances in certain relevant fields, both for development tools and for final system implementation. We believe that to narrow the gap between the challenges in developing trustworthy next-generation autonomous systems and the present state of the art, the research and industry community must construct a common engineering foundation for developing such systems. This foundation, which we term "autonomics,"* should address the unique challenges

Significance

Autonomous systems are replacing humans in a variety of tasks, and in the years to come, such systems will become central and crucial to human life. They will include vehicles of all kinds, medical and industrial robots, agricultural and manufacturing facilities, traffic management systems, and much more. While many organizations strive to develop the next generation of trustworthy, cost-effective autonomous systems, a major gap exists between the challenges in developing these and the state of the art. There is a crucial need for a common scientific and engineering foundation for developing these systems, which we term "autonomics." We believe that such a foundation will dramatically accelerate the deployment and acceptance of high-quality autonomous systems, for the benefit of human society.

Author contributions: D.H., A.M., and J.S. designed research, performed research, and wrote the paper.

Reviewers: C.K.C., Iowa State University; W.D., Carl von Ossietzky University of Oldenburg; and M.Y.V., Rice University.

Competing interest statement: D.H., A.M., and M.Y.V. are coauthors on a prior conference proceedings article [D. Harel, A. Marron, A. Rosenfeld, M. Vardi, G. Weiss, *Proceedings of the 33rd AAAI Conference on Artificial Intelligence* (AAAI-19), (2019) 33: 9770-9774].

See QnAs on page 17465.

[1]To whom correspondence may be addressed. Email: dharel@weizmann.ac.il.

First published July 21, 2020.

*The term autonomics has been used in the past in certain circles in the context of autonomic biological systems as well in studying self-regulating systems. We believe that with the emergence of ubiquitous autonomous human-made systems that also interact with humans and nature, the term can be used in a significantly broader sense, to incorporate the study, the underlying principles, and the development of autonomous behavior—both biological/natural and engineered.

relevant to such systems, by providing concepts, perspectives, and engineering principles, as well as the supporting methods and tools. The concepts should be broad enough to encompass multiple ways of doing things, and will have to include means for selecting among alternative designs, technologies, and tools.

We believe that the availability of such a foundation has the potential to dramatically accelerate the deployment and acceptance of high-quality, certifiable autonomous systems, built for the benefit of human society.

Next-Generation Autonomous Systems

Preliminary Definitions. Over the years, many definitions have been offered for autonomy (e.g., refs. 11–13). "Autonomic computing" (e.g., ref. 14) focuses on systems capable of self-management, and in particular, automating dynamic configuration. The research area of agent-based design and in particular "multiagent systems" (e.g., ref. 15) offers a perspective of autonomy in paying special attention to the issue of combining local goals with collaboration rules and distributed algorithms to achieve system-wide overall goals. Autonomy is often associated with "self-awareness" (e.g., ref. 16), which implies the system's ability to perceive changes of the environment and use "knowledge" of its own states to react adequately, so that a set of goals is achieved. "Symbiotic computing" (e.g., ref. 17) studies how autonomous systems can interface and collaborate with humans and with complex organizations, considering the many technical, commercial, and ethical implications thereof. Also, controllers designed with patterns like Sense–Decide–Act or Monitor–Analyze–Plan–Execute (MAPE), are sometimes referred to as being autonomous; however, such control loops alone are clearly not sufficient for a system to move around, conduct itself, and function correctly in arbitrary real-world open environments.

In order to streamline the ensuing discussion, we offer, in this subsection and in the next one, definitions for some basic concepts (see also refs. 18–20), and illustrate them with an example of an AV for factory-floor and plant-yard deliveries (termed here FFAV).

"Systems" are the artifacts that development teams are out to build. A system works within, and reacts to, an external environment, and it consists of two types of components, agents and objects (the latter, as explained below, are typically not built by the development team), which, as we shall see, operate within a common internal system environment. The coordinated collective behavior of the system's agents and objects is designed to meet some global, system-wide goals.

"Objects" are those components whose programmed behavior is not affected during system development. For example, objects of the FFAV system may include ready-made components, such as the motor, a set of cameras, or a steering mechanism whose input is, say, the desired angle of the front wheels. A system often interacts with objects that are not part of it but are part of the environment. Such are, in the case of the FFAV, machines on the factory floor (which may be mere obstacles or recipients of deliveries), and packages to be delivered. Objects have "states," which can be changed by agents or by other objects, or can change "spontaneously," for internal reasons. These internal and external objects become part of the system's internal and external environments, respectively.

"Agents" are the main behavioral elements of an autonomous system. They are those designed (programmed, built) as part of the system's development process.[†] Agents have "agency": They are proactive and pursue specific goals that may change dynamically. Agents can monitor objects from the internal and external environments and can change their states. They can also coordinate their own actions with other agents. Thus, the FFAV system may either have a single agent for all its functions, or separate agents for different tasks, such as work scheduling, route planning, travel control, gripping, and carrying. Agents can themselves be autonomous systems, and can, in turn, hierarchically contain other agents and systems.

The "internal environment" is the lower-level physical and virtual infrastructure used by the system's agents and objects. It may include the computer/processor/memory, batteries and other power sources, the operating system, communication hardware and software, and database management software.

The "external environment" of a system (often simply called the "environment") is the collection of all entities with which the system might interact. It may include other systems (with their objects and agents) and stand-alone objects, and any other physical or virtual entities that may affect, or be affected by, the system's behavior. Key entities in any system's external environment are of course humans—with their unpredictability, initiative, and power and authority to modify system behavior.

Defining Autonomous Behavior. We say that a system or an agent (for simplicity, we shall stick to system below) manifests "autonomous behavior" if it embodies the following five behavioral functions, which are carried out with little or no intervention from humans or from other systems.

Two functions are combined to enable the system to build for itself a useful representation of the state of the external environment. "Perception" is the function that inputs stimuli, interprets their basic meaning, and removes ambiguity, yielding relevant information. Often perception has to deal with multimodal inputs, such as vision, sound, heat, touch, radar, and data communication from other systems, all obtained using mode-specific sensors and input devices, and has to then amalgamate the received information. The second function is "Model Update," which uses the information provided by Perception to create and constantly update an integrated run-time model representing the system's environment and its states. This model will then be used in ongoing decision making.

Two other functions constitute the system's adaptive decision process. This means that decisions consider many possibly conflicting goals, in a way that depends on the system's current state and that of the environment. "Goal Management" chooses from among the set of goals the ones that are relevant to the current state. "Planning" computes a plan to achieve the set of goals produced by Goal Management, subject to state-dependent constraints; this is the agent's action in response to the current environment state, and may consist of a sequence of commands to be executed by actuators. Both Goal Management and Planning may take part in addressing conflicting goals; they do so by, for example, prioritization, constraint resolution, proceeding on multiple concurrent plans (deferring the decision), and consultation with humans or with other systems.

The fifth function that characterizes autonomous behavior is "Self-Adaptation," which caters for dynamic adjustment over time of the system's goals and the goal management and planning processes, through learning and reasoning, based on the evolving state of the system and its environment. Such adaptivity could come in many forms: very near term, e.g., using trial and error and recent experiences to go around an unfamiliar and not-previously specified obstacle; "life-long learning" of the system, constantly reevaluating its entire history of sensor information, its actions, and its successes and failures, in order to better achieve its goals in a dynamic unpredictable environment; even achieving certain independence, where the autonomous system studies the entire changing environment, far beyond its task scope, and is able to adjust not only its operation, but also its

[†]We ignore here the question of whether a given component, already developed and then incorporated "as is" into the system, should be considered an agent or an object thereof. Similarly, we sidestep the question of whether agents that are part of systems in the external environment, like those of autonomous manufacturing machines in the context of the FFAV, should be considered agents or objects.

goals (consider an AV that upon discovering that a key user cannot travel, arranges on its own to fetch people or objects to that user, or facilitates a teleconference). The question of when should system adaptivity be curtailed and humans should be called upon is a broad one, and requires much thought and further research.

Next-Generation Autonomous Systems Are Different. Next-generation autonomous systems, both those that are already beginning to emerge and definitely those of the future, differ from existing systems in the several key aspects. These aspects, which we now discuss, position the "next generation" adjective we attach to the systems we are talking about here beyond the realm of the ones that are being developed now and will be operational in the very near future. Rather, our subject matter are those systems we feel will be prevalent only a good number of years henceforth.

They have a large variety of possibly conflicting system goals. A typical next-generation autonomous system will not be focused on a small number of well-defined goals, such as winning a game of chess, or a vehicle reaching a destination without collisions. They will typically face a far wider and more elaborate set of goals, as humans often do. Consider, the FFAV making a highly critical (and expensive) delivery, which may be at risk due to a safety issue. The situation is further complicated by the financial and legal considerations of its manufacturer.[‡] For example, the owner of a chemical plan that uses an FFAV may want to allow the FFAV to (carefully) disobey a stop sign when making a very urgent delivery, but the FFAV's manufacturers might have programmed it for absolute compliance with the law, in order to reduce their liability.

Their environment is dramatically less predictable. Even autonomous systems of the present already have to deal with an enormous number of known environment configurations, and those we do not know about yet will obviously add a whole new order of magnitude to this difficulty. While an AV's handling of varying road topologies and traffic volumes and speeds can probably be addressed using existing technologies, there are more complex issues, which humans handle routinely and which are still not adequately addressed. For AVs, these include, e.g., the whims of bicycle and motorcycle riders weaving in and out of traffic on roads, sidewalks, and crosswalks; police instructions, spoken or signaled; poorly marked temporary diversions; emergencies that have not yet been handled by first responders, such as a traffic accident, a landslide/rock-fall or flooding; or an urgent request by a passenger to stop and step out, but where there is no safe place to do so. These kinds of difficulties are caused by the increased dependency on hard-to-predict physical aspects of the dynamic environment, compounded by the increased mobility, distribution, and sheer multitude of systems.

They require rich interaction with humans. Classical human–computer interaction (HCI) is typically geared toward trained users or operators controlling automated tasks. Future interfaces will have to deal with the overall behavior of the system as sensed by humans, with the way the system affects human behavior and with the way humans think about system behavior. Next-generation autonomous systems will affect and put at risk a far wider circle of people, and will be increasingly exposed to both helpful and adversarial human actions, with the required communication and interaction that they necessitate. Suffices to think of a traffic jam caused by an AV on a busy highway, an autonomous crane on a busy construction site in a city center, or a medical supply robot, scurrying down a crowded hospital corridor to make a life-saving delivery.

The issue of interaction with humans goes much deeper than classical HCI. First, because future systems will operate in common human environments, having to interface with humans who are neither users nor operators, and over whom the owner of the autonomous system has no control. Their behavior will not only have to be functional, efficient and safe, but will also have to appear to be so, in order to instill in humans the confidence that this is indeed the case. These systems will be interfacing with humans in wholly new ways, even as part of normal everyday routines, such as negotiating the right-of-way through an office door, or pointing out a spill on the floor to a passing robotic cleaning assistant.

Second, the human–computer interface itself will have to be far more extensive than a mere display and keyboard. It will encompass much of what the autonomous system understands and does. If the FFAV has to hand a fragile package to a human, and take from them another package, how does it communicate its readiness to hand over one and receive the other, its questions (e.g., has the human recipient already secured a hold on the first package?), its state (e.g., that it is now holding the second package safely so that the human's hold and attention is no longer needed), and so on.

Third, special attention has to be given to those parts of the interface that allow a human to interrupt the operation of the autonomous system or change it abruptly. If a worker just dropped a contact lens that the FFAV cannot see, how does he/she immediately stop it? If the FFAV was given the wrong package, how does the human call it back? If some emergency work blocks the normal, preprogrammed route of the FFAV, and a detour cannot be easily discovered, how does one give the FFAV an alternate, ad-hoc instruction, in real time and in a natural way, that will cause it to use a particular alternative route, say, the handicap ramp behind building C?

Why a New Foundation?

Our main claim in this paper is that developing trustworthy next-generation autonomous systems requires addressing fundamental issues that have not been dealt with adequately by present research or industrial experience. We call upon the research and engineering community to create and evolve a foundation for developing such systems, which will recommend engineering practices and methods, point at tools and technologies, and offer open-source bases and examples. It will also include meta-information, such as reliable means for selecting among system design and development alternatives. While this autonomics foundation should touch upon all aspects of system engineering, initially it should not aim at rewriting well-accepted system engineering principles, but address the "burning" issues (such as those we discuss in the upcoming sections) and propose ways to deal with them throughout development. Otherwise, a broad, all-encompassing effort could dilute and obscure the important innovations for which it was created, and it may even be completely dismissed as a futile effort to "boil the ocean." At the same time, should additional new approaches to ingrained practices emerge, they should be seriously evaluated and incorporated where applicable.

The existence of gaps between the state of the art and achieving the desired trustworthiness has been articulated, e.g., in the Institute of Electrical and Electronics Engineers white paper (21) and in Neumann (22). Specifically, the latter focuses on the need to handle the impact of component vulnerability on full, composite system vulnerability. The literature dedicated to building and testing complex, autonomous, safety-critical systems (e.g., ref. 23) provides precious little in way of theories and tools for ensuring one's confidence (or trust) in the system's run-time behavior in face of unpredictable situations. In a closely related

[‡]Requiring the ability to handle possibly conflicting goals might intersect with the thorny distinction between the two categories of artificial intelligence that are sometimes termed "weak AI" and "strong AI" (https://en.wikipedia.org/wiki/Artificial_general_intelligence and https://en.wikipedia.org/wiki/Weak_AI). We believe that the decisions faced by next-generation autonomous systems, complex as they may be, still do not require resolving all of the issues around general, human-level artificial intelligence.

discussion of the challenges facing next-generation systems, Chang (24, 25) argues that new solutions, and indeed a paradigm shift, are both required and possible, and that these can be enabled by new approaches to situation analysis.

To reinforce our argument about the importance and extensiveness of the required foundation, we shall focus here on one central aspect, which is at the very heart of autonomous system engineering—the decision making. We present three partially overlapping challenges in developing decision-making processes, for which satisfactory solutions have yet to come.

Challenge I: Specifying Behavior. Behavioral specification is needed in virtually all stages and activities of the development process: requirements, design, simulation, testing, verification, and validation. Behavior should preferably be specified in some rigorously defined language with agreed-upon dynamic semantics, but, at the very least, can also be done in a way that can be mapped directly to precise and technically oriented natural language descriptions. Although numerous diverse computer languages have been developed for this purpose—procedural, declarative logic-based, state-based, scenario-based, and more—we argue that in next-generation autonomous systems the very specification of behavior introduces new issues that call for extensive research. Such specifications should cover both classical desired and undesired sequences of events and actions, related goals and their mutual relationships, actual behavior, as observed and interpreted by machines and by people, and inferred behaviors that may have not been fully observed. The specification should serve to build systems, to test them, to enable communication between systems, between humans, and between systems and humans, and to enable the meta-analysis of such specification.

When it comes to complex autonomous systems, the specification of even a single simple goal is hard. Assume that, for the first time, an employer wants to completely automate the floor-cleaning process. What kinds of specification are we after? Should it be focused on actions (e.g., where and how to sweep), on environment objects and entities (e.g., what kinds of dirt should be removed, and from where), or on states and results (e.g., what should the floors and shelves look like once the job is done)? How should one tell developers (and the system) about the need to move small objects or unplug devices that are in the way, or about dealing with such risks as breaking something?

We believe that what we need here are ways to describe the relevant "world" and its associated behaviors. For this, we propose to develop domain-specific ontologies of objects, properties, actions, and relations. This direction may extend or learn from current ontologies like the CYC project (26), Google's Knowledge Graph, the OWL web ontology language and others, but may also take on different design directions. For example, the very essence of entities may be associated with action-related information about what these entities do, what can be done with them, or how other systems are expected to react thereto. The goal in this case is not to achieve comprehensive (yet elusive) knowledge, but to enable decisions within the system's action repertoire. For example, if the FFAV sees in its path an object it cannot recognize, it should be able to present a picture and perhaps sensor information to other facilities, such as a server in the cloud. The server might then inform the FFAV as to what the object is, so it can apply its existing rules, or it may instruct the FFAV to take certain actions based on the server's knowledge and logic.

In the context of autonomous systems, some progress along these lines can be found in, e.g., traffic sequence charts (27), US National Highway Traffic Safety Administration scenarios (28), autoware software for AVs (29), and open source simulators like CARLA (3). Each of these uses its own terms and concepts as building blocks in a bottom-up tool construction.

Beyond the issue of specifying single goals lies the extreme difficulty of specifying how the system should balance, prioritize, or weigh several, often competing goals under a bewildering multitude of circumstances. Even in a single given situation, and even if we allow the use of natural language for specification, it is often almost impossible to state what the system should or should not do. Many future generation autonomous systems will have to make complex decisions involving major human and business risks, and we doubt that stakeholders can prescribe in advance what the system should do in each case.

Furthermore, of course, in addition to such technical issues of specification, there are also weighty ethical issues. Courts of law rule on whether a decision made by a human was right or wrong, negligent or not, in line with what is expected of a reasonable person. This will become significantly harder in the realm of autonomous systems, which, for example, will have to determine where exactly to make the initial incision during a surgical procedure, or to decide in a split second between two very bad alternatives in an emergency driving situation.

During simulations, stakeholders often encounter emergent properties and unexpected behaviors that were not mentioned in the development process. For example, when observing the behavior of an AV, such as an FFAV or a golf cart, one may notice that it sometimes repeats a path with unusual precision, creating unexpected wear on the floor, ground, or grass. New requirements may be desired as a result, such as randomizing paths, documenting a limited set of supported surfaces, or even taking advantage of this kind of predictability in other parts of the system.

Dynamic changes in specifications constitute yet another complicating aspect. It is not unusual for a human operating on a certain task to receive new information or instructions about changes in goals, in means to achieve them, or in assumptions about the environment. Humans most often deal reasonably well with such updates. Autonomous systems will have to support such modes of communication and of reactive behavior (16).

Making this happen is not easy. It is not even clear how to specify a formal version of a "trouble ticket," which describes an event, property, or pattern that was noticed by the human observer but was not part of the original specification. Furthermore, one would want to automate the detection and articulation of emergent properties, since testing and simulation are likely to be highly automated, with limited opportunity for human observation. For anticipated behavior (desired or undesired), this would be quite similar to testing; but, for unexpected behavior, automating the capturing itself in a formal, yet succinct and intuitive way, would appear to be a major challenge, requiring substantial extension of current research in fields such as specification and mining anomaly detection (see, e.g., refs. 30–32). In fact, we believe that Knuth's famous quote, "Beware of bugs in the above code; I have only proved it correct, not tried it," goes beyond recognizing the importance of testing given the limitations of formal methods and correctness proofs. It can be used to support our belief that testing is a must also because just observing the system in operation yields totally new insights about what the system does and does not do, and what it should or should not do.

Explainability and interpretability are especially relevant to emergent properties, particularly for those parts of the solutions based on neural nets and other "black-box" approaches. In a way, explanations induce a model on the seemingly model-less machine-learning solution. Additionally, here too, summarizing such execution patterns automatically is a challenging problem that is the subject of active research. We discuss this issue somewhat in *Challenge III: Combining "Model-Based" and "Data-Driven" Approaches* below.

The task of specifying and explaining behavior that must dynamically reconcile multiple goals can be aided by initially adding weights, priorities, and mutual constraints to goals, as well as

just observing the system in operation and endowing the components responsible for the various goals with dynamic internal negotiation capabilities. A key role will also be played by mechanisms for specifying and handling contingent behavior, which reacts to and handles negative conditions directly related to the system's actual behavior. The ability to provide concise explanations of the system's decisions, both in real time and after the fact, will be of great value, allowing developers, and the system itself, to judge the programmed decisions and adjust them as needed.

Challenge II: Analysis. By analysis, we mean simulation, testing, formal verification, and system validation against the tacit needs of the stakeholders (STV&V for short). While these techniques will be of paramount importance for next-generation autonomous systems, it is well known that none of them provides complete assurances even for current systems, so they will have to be used in ways that complement each other.

The various techniques comprising STV&V all involve one manner or another of executing a system or a model thereof in a controlled fashion, and/or traversing or analyzing the resulting states. Simulation is perhaps the most "hands on" of these, and facilitates observing emergent behaviors—desired, undesired, and not yet specified—under a variety of conditions. However, the simulated environment will always be an abstraction and simplification of reality. There are numerous simulation tools relevant to AVs; see, e.g., refs. 4, 8, 11, and 17. While these are effective and provide important features—albeit, spread across different tools—the foundation proposed here calls for additional important capabilities, such as far greater user control over environment variability and the ability to automatically detect and evaluate new emergent properties that were not in the original test specifications.

One of the main reasons that satisfactory STV&V analysis calls for foundational work, is the vast number of objects and variables involved in complex autonomous systems, and the even greater number and intricacy of the interactions between them. This is what we now discuss.

Autonomous systems will typically have to deal with numerous new elements, which are often ignored or simplified, or are controlled by other systems. Just think of an FFAV deployed at a busy outdoor factory yard, with people, equipment, and vehicles moving around, possessing distinct shapes, colors, reflection types, textures, sizes, locations, positioning, and routes. Faithful simulation and effective testing of these systems and environments is a formidable task. Furthermore, as to interactions thereof, here the problem becomes alarmingly worse. The system's "intentional" activities may be reasonably controlled, but the number and complexity of the possible interactions and indirect effects between all objects in the environment and the system, are mind-boggling. Consider testing a very small FFAV working its way through a throng of humans and machines, perhaps even other living creatures—as in a livestock show. The interactions it has to deal with are not limited to the obvious goals of, e.g., reaching a destination while avoiding collisions and abiding by traffic laws. How about the interactions between objects that arise when the FFAV tries to avoid splashing passersby when it crosses a puddle? How should it deal with cases where it might inadvertently startle humans when quietly and suddenly showing up at their side, or when it gets entangled in a cable or in loose fabric?

As stated earlier, the fact that humans inhabit the external environment of autonomous systems further complicates matters. For example, a human's ability to learn and adapt to new conditions, is likely to surpass that of most systems for a long time to come. Such human adaptivity may be relied upon for certain systems, where developers might allow systems to modify their own behavior even without first coordinating with affected humans, trusting that the humans will "figure things out" on their own. This adaptivity is also what often allows humans to correct or override the behavior of systems they control. Either way, it is useful to analyze the boundaries of environment assumptions, by taking into account also the whims of humans.

As we did for the specification challenge, we list below some of the issues that the autonomics foundation should address regarding analysis. One, which is a precondition to any kind of analysis, has to do with the "modeling of environments." We envision using domain-specific libraries for various kinds of systems and tasks, in order to deal with the physical three-dimensional space of real-world objects and their mobility. These libraries will be different for different application areas. Just think of the very different kinds of environments relevant to a medical system and a transportation system. The environment would have to be modeled using languages and tools that are able to describe knowledge of the environment and assumptions thereof, and to achieve a desired level of realism by controlling abstraction levels and simulation granularity.

The second issue the foundation will have to address, which is particularly important for analysis, involves the "infrastructure" needed for STV&V. This would have to include mechanisms that orchestrate and control executions; set up the physical or virtual environments; play the role of the environment when needed; observe, record, analyze, and act upon the system's actual behavior; and interface with engineers for these and related functions. We want to be able to test and simulate autonomous agents in interaction with the complex cyber physical environment for which they are being built. The infrastructure should be "state-aware" and transparent, being able to communicate with engineers using natural interfaces and logs that describe intuitively the state of the external environment, the internal state of the system and its agents, and the state of agents' perception of the external environment.

For example, assume you want to figure out what the FFAV will do when it faces an obstacle consisting of two posts placed at a distance apart that is barely more than the width of the FFAV. Will it move between them or bypass them? Standard testing and simulation techniques call for actually placing the obstacles and observing the system's behavior. However, unless we also carefully check the feedback from the tested system as to what it "thinks" it saw, one cannot be sure if it perceived the conditions as intended. Thus, the FFAV may indeed pass between the posts, but for the wrong reason: It might have misclassified one or both of them.

Such perception control is a particular case of state awareness, where during STV&V the infrastructure is able to report the system's state and that of the environment; can guide the operation of system components based on the states of others; and can report, and react to, things the system does and does not do, its execution paths, etc. The complexity of all this is amplified by the unpredictability of behavior. Even the relatively simple problem of determining which of the agent's states and interactions may occur in parallel with which others is extremely difficult.

The third major challenge of the analysis part of the foundation has to do with controlling and measuring the "behavioral coverage" achieved via testing, whether virtual/in silico or by deployment of the autonomous system in the real physical world. Such coverage refers to the space of all composite systems states, across multiple components, as well as of the paths and scenarios for reaching these states. Current testing methods focus on a variety of relevant aspects, including coverage of development entities (like statements, components, program changes, and requirements/assertions), automated test execution and evaluation, and automated generation of scenarios (33, 30). Makers of AVs sometimes present the number of miles (real and

simulated) that their products have driven (see, e.g., refs. 34 and 35) as an indicator of behavioral coverage.

For the kinds of systems we have in mind, which will be deployed in large numbers worldwide, these approaches are inadequate. Even what appears to be the bare minimum here—a practical approach to measuring overall composite state coverage for both system and environment—is already hard enough (see discussion in ref. 36). We envision having to develop techniques to automatically generate rich sets of scenarios, subject to criteria that can be external, i.e., from the environment and the real world, or internal, such as intricate behavioral combinations of specification and implementation entities. Furthermore, given the inability to exhaustively cover all run-time possibilities, we need support for accelerated metamorphic testing in physical environments; i.e., checking thoroughly that the system behaves correctly for a given scenario, and then quickly providing assurances for many other scenarios that differ from the basic scenario only by small physical changes. We also need fitting criteria for evaluating the testing process itself.

Finally, the autonomics foundation will have to address "formal verification." Even the best current verification methods can be used successfully only for single components or for greatly simplified models of the entire system. Also, not only is the behavioral specification of the system itself very hard, but it is no easier to specify the assertions that describe the behavior we want to verify in terms that are readily aligned with the expectations of the human users and engineers. This is further complicated by the fact that whether some behavior is desired or not may not by a binary decision but a quantitative one, spanning multiple scales (37).

Since many next-generation autonomous systems will have components based on machine learning (ML), the formal verification of neural nets, and the ability to supply adequate explanations of their internal behavior, will become increasingly important. These problems are long recognized as being very difficult, and there is an emerging field of research around them, whose initial results look very promising (38, 39). Accordingly, the next section takes a closer look at the challenge involved in incorporating such AI-based learning techniques into the foundation.

Challenge III: Combining "Model-Based" and "Data-Driven" Approaches. In the ensuing discussion we use the term model-based to include classical software development approaches, all of which employ traditional programming languages and prescribe step-by-step processes and/or rules that are meticulously handcrafted and organized by humans. This also includes model-drive engineering (MDE) techniques that use languages like those that have been made part of the broad unified modeling language (UML). However, we do not restrict ourselves to MDE. We have decided to use the term model-based in order to emphasize the fact that the designer is required to build and provide a thorough technical description (a model) of the problem, its inputs, its outputs, and the required processing and behavior, in terms that are aligned with the problem domain.

In contrast, we use the term data-driven to encompass all techniques that involve ML (including, but not restricted to, deep neural networks), statistical analysis, pattern recognition, and all related forms of computing in which the system's behavior is derived from supervised or unsupervised observation. The latter can include observing input and output events and occurrences in the real world or in the processing of other systems, even that of earlier versions of the system under development. The desired behavior of the system we are developing is thus inferred; not prescribed as in model-based approaches.

There is a growing call to find ways to combine the two techniques, leveraging their relative advantages to complement each other (40–42). Nevertheless, there is still no agreement on how to do this, the combination being very different from integration practices in classical engineering. In addition, of course, due to the new challenges involved, the problem is further exacerbated for next-generation autonomous systems.

There are several differences between traditional software development and constructing solutions based on ML, which must be taken into account when trying to integrate the two. To better concentrate on the integration issue in this subsection, we disregard the still-open research problems in each of them.

The first difference involves the "general life cycle." Traditional software engineering—in any of a number of classical life cycle methodologies—calls for requirements elicitation and specification, design, code, testing, and so on. In contrast, developing a module or system based on ML involves totally different stages, such as the collection, validation, and sampling of training data, the actual training, evaluation, revision, and retraining, etc.

The second difference concerns "specifying requirements." Consider even very simple cases, such as requiring that an electrical switch must turn off when the temperature reaches 80°, or that the brakes must be activated when a stationary obstacle is sensed and the stopping time at the current speed is less than 1 s. These are well defined, and engineers can translate them easily into working components, but for a system trained to handle excessive heat or avoid collision based on positive and negative examples, it is not at all clear how to use the requirements or how to incorporate them into the respective ML components.

Related to specifying requirements is the issue of after-the-fact "explainability" (also referred to "interpretability"). This calls for the ability to justify, or rationalize, a particular system decision using problem-related parameters and arguments. More generally, we want to be able to describe in this way what the system does and the underlying rules, algorithms, and computations it uses. Despite the remarkable success of neural nets in performing many kinds of tasks, their internal workings are often a mystery. Current ideas addressing this problem are still a far cry from the situation with traditional programming, for which engineering practices recommend producing code that is easy to read and understand, and to enhance it with ample comments. Moreover, even if explainability and interpretability tools are eventually able to extract the tacit rules behind the operation of large neural nets, the way these rules relate to the net's actual mechanisms will be very different from the relation between natural language descriptions and source code in classical programming languages.

In addition, the difficulty in specifying the behavior of neural nets and explaining what they do and why they do it, makes it very difficult to analyze their behavior. While some initial work has been done on checking properties thereof (see, e.g., ref. 39), there is still much to be done on the testing and verification of systems based on learning.

An important related difference involves "decomposability," which is crucial in most stages of development, e.g., for understanding and anticipating system behavior, finding and fixing errors, carrying out enhancements, and assessing the impact of changes. In model-based designs, most system artifacts can be hierarchically decomposed into well-understood functional and structural elements, the role they play in the full system being more or less clear. In contrast, the design of data-based ML solutions is typically accompanied by an end-to-end mindset—system-based or problem-based. Being able to decompose a ML solution into meaningful parts appears to be an interesting challenge, which will, of course, bear upon explainability and verification.

Finally, we mention the differences between the two approaches with regard to their "trustworthiness and certification," which are clearly related to testing and verification. Many kinds of autonomous systems are highly critical; failure to meet their expected behavior can be disastrous. Critical system design calls for providing appropriate trustworthiness guarantees for the system's functions and reliability. These are often specified in standards, like DO178B for avionic systems and ISO 26262 for electronic components in the automotive industry. In principle,

model-based techniques give rise to predictability at design time, but components based on ML are not engineered in the same way. Achieving for them an accepted level of trustworthiness and certification requires wholly new technical solutions, since conventional testing and simulation techniques are inadequate given the complexity and unpredictability associated with next-generation autonomous systems. Complementary nontechnical measures, such as risk management, the concept of insurance, or the use of the justice system as deterrent against negligence, are separate issues altogether and are beyond of the scope of this paper.

These significant differences illustrate the magnitude of the methodological and technical integration challenge that the autonomics foundation will have to address. To give a relatively straightforward example of this, consider a proposed system that is to ultimately consist of conventional model-driven components (based, e.g., on an object model, algorithms, scenarios, rules, and decision tables) and ML data-driven components (based, e.g., on neural nets). At some point, the engineers will have to decide which subproblems should be solved using which of the two approaches. Sometimes the answer is easy: For example, reading traffic signs can be fully ML based, while the decision to remain below a known speed limit can be model based. Staying in lane on a clean, well-marked highway might be model based, while negotiating a road surface covered with sand could very well be ML based. However, in many cases, the choice between the two approaches will be a lot more difficult. Besides, designing the run-time decision process that determines which situation is the relevant one, in order to activate the proper component (e.g., whether the road surface is clean or sandy), is in itself a design challenge; the parameters of the decision may not be clear-cut, and the designers will have to choose whether to use a model-based, data-based, or hybrid approach for this subtask.

The foundation should describe and discuss various specific approaches to the integration. One example is a "pipeline" approach, where some preprocessing is done by one method, and interim results are passed on to the other method. In the "divide-and-conquer" approach, the problem is divided into subproblems, and different techniques can be applied to each; the actual division can itself be model-based or data-driven. In another approach, one method (usually rules) serves as a "protective wrapper" around the other (usually ML-based), constraining the latter to be within some decision domain. In additional variants, an approach that is mostly model based can incorporate several data-driven black boxes to enrich sensor processing or to solve particular subproblems. One can also apply both techniques to develop complete solutions, thus creating redundancy, and then use various composition or "voting" techniques to yield the final system behavior.

By definition, such approaches for composing hybrid solutions from model-based and ML-based components will also imply the first level of the decomposition of the solutions.

Discussion

Next-generation autonomous systems are definitely going to be built and will become commonplace in the years to come. Many of them will manifest some important form of criticality. They will have to cope with the uncertainty of complex, unpredictable cyber physical environments, and will have to adapt to multiple, dynamically changing, and possibly conflicting goals. They will be expected to collaborate harmoniously with humans, giving rise to so-called "symbiotic" autonomy. Their predicted advent reflects the transition from "narrow" or "weak" AI to "strong" or "general" AI, which cannot be achieved by using just conventional model-based techniques or ML alone. Thus, classical software and systems engineering will have to be thoroughly enhanced.

AVs provide an emblematic topical case, illustrating the challenge. For example, due to the lack of standards and compliance assessment techniques, some public authorities allow self-certification for AVs, despite their criticality. Another issue is evidence. Manufacturers will often publicize only partial information about their testing, such as the distance an AV has been test-driven. One can then only hope that the behavioral coverage was indeed sufficient, and, that someone other than the manufacturer indeed examined the tests and deemed them satisfactory. Another trust-related issue is the fact that critical software can be updated regularly, which creates the hope that certain kinds of failures of an AV would be immediately corrected in all AVs worldwide. However, this also raises the concern that updates might be deployed with less-than-adequate testing, causing problems of more critical impact than standard updates to the operating systems of smartphones or personal computers.

All this has generated lively public debates. Many important voices tend to minimize the risks from the lack of rigorous design methods: Some claim that we should accept the risks because the benefits will far outweigh them. Others accept the empirical methods and argue that rigorous approaches to complex problems are inherently inadequate. Some people are overoptimistic, arguing that we really do have the right tools, and it is just a matter of time. In addition, besides all of this, we must take into account all relevant ethical/moral, legal, social, and political issues, a vast topic that is obviously outside the scope of this paper.

Our paper's contribution is twofold. First, we propose a basic terminology for next-generation autonomous systems, and a framework capturing their main characteristics, and discuss how they differ from present-day autonomous systems. The framework provides insights into the spectrum of possibilities between automation and autonomy, and is intended to help in understanding the degree of autonomy of a system as the division of work between a system and human agents.

Second, we claim that the advent of next-generation autonomous systems raises an extraordinary scientific and technical challenge, and advocate the need for a new foundation that will address the key open issues in their engineering. Such an autonomics foundation will hopefully lead eventually to trustworthy hardware/software systems. The degree of success in meeting this challenge will ultimately help determine the extent of acceptance of such systems, as a compromise between their estimated trustworthiness, the anticipated benefits of the automation they afford, and the required changes in other systems and in human behavior.

We anticipate that forming this foundation will require major and ground-breaking efforts in the three main directions presented above, and which we briefly summarize below.

The first is to develop a rigorous theory and supporting tools for dealing with heterogeneous specifications. These should make it possible to characterize system behavior in a broad fashion, including the behavior of its individual agents, as well as the system's global behavior in terms of its overall goals and its emergent properties.

The second direction is aimed at providing sufficient evidence of a system's trustworthiness. We have emphasized the paramount importance of modeling and simulation: We need faithful, realistic modeling of behavior, as well as semantic awareness, so that the experimenter has access to a meaningful abstraction of the system's dynamics, allowing controllability and repeatability of the testing. The latter will also allow behavioral coverage, where we measure the degree to which relevant system configurations have been explored.

The third direction of the effort required for the foundation involves adopting a powerful "hybrid" design approach, seeking trade-offs between the trustworthiness of classical model-based approaches and the performance of data-based ML ones. Taking better advantage of each approach requires the development of common architectural frameworks that would integrate modules characterized by their pure functionality, independent of their design approach. Developing the theory for decomposability, interoperability, and explainability of data-based modules is essential for reaching this goal.

In summary, we are at the beginning of a revolution, where machines are called upon to progressively replace humans in their capacity for situation awareness and adaptive decision making. This requires some aspects of general AI, which go beyond the objectives of ML-enabled intelligence. The extent to which we will ultimately use and benefit from autonomous systems will depend on how much we trust them, an issue which is the main reason for writing this paper.

Data Availability. This paper is not accompanied by any data, protocols, code, or materials.

ACKNOWLEDGMENTS. We are grateful to Guy Katz and Orna Kupferman for valuable discussions in the early stages of preparing the paper. This work was supported in part by grants to D.H. from the Israel Science Foundation, Intel Corporation, and the Estate of Emile Mimran, and from his endowed William Sussman Professorial Chair of Mathematics at the Weizmann Institute.

1. Boston Dynamics, "Boston Dynamics Spot Robot is ready to leave the nest" (2019). https://www.youtube.com/watch?v=jEr1kRf_FC0. Accessed 6 July 2020.
2. S. C. Niquille, Regarding the pain of SpotMini: Or what a robot's struggle to learn reveals about the built environment. *Architectural Design* **89**, 84–91 (2019).
3. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator. arXiv:1711.03938 (10 November 2017).
4. R. Majumdar, A. Mathur, M. Pirron, L. Stegner, D. Zufferey, PARACOSM: A language and tool for testing autonomous driving systems. arXiv:1902.01084 (4 February 2019).
5. Hexagon AB, MSC Software Virtual test drive. https://www.hexagonmi.com/products/computer-aided-engineering-cae-software/msc-software. Accessed 6 July 2020.
6. Cognata, Inc., Cognata driving simulator. https://www.cognata.com. Accessed 6 July 2020.
7. DEEL, AI project. https://www.deel.ai/. Accessed 6 July 2020.
8. SafeTRANS, SafeTRANS project. https://www.safetrans-de.org/en/. Accessed 6 July 2020.
9. US Department of Transportation, Preparing for the future of transportation: Automated vehicles 3.0. https://www.transportation.gov/sites/dot.gov/files/docs/policy-initiatives/automated-vehicles/320711/preparing-future-transportation-automated-vehicle-30.pdf. Accessed 6 July 2020.
10. HumanDrive, HumanDrive Project. https://humandrive.co.uk/. Accessed 6 July 2020.
11. M. Luck *et al*., "A formal framework for agency and autonomy" in *Proceedings of First International Conference on Multiagent Systems* (AAAI Press, 1995), vol. 95, pp. 254–260.
12. S. Franklin, A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents" in *International Workshop on Agent Theories, Architectures, and Languages*, J. P. Müller, M. J. Wooldridge, N. R. Jennings, Eds. (Springer, 1996), pp. 21–35.
13. SAE International, Automated driving levels of driving automation are defined in new SAE International standard j3016. https://cdn.oemoffhighway.com/files/base/acbm/ooh/document/2016/03/automated_driving.pdf. Accessed 6 July 2020.
14. IBM, "An architectural blueprint for autonomic computing" (Autonomic Computing White Paper, 2006). https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf. Accessed 6 July 2020.
15. O. Shehory, A. Sturm, Eds., *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks* (Springer, 2016).
16. S. Kounev, J. O. Kephart, A. Milenkoski, X. Zhu, Eds., *Self-Aware Computing Systems* (Springer, 2017).
17. S. M. Dambrot, D. de Kerchove, F. Flammini, W. Kinsner, L. MacDonald Glenn, R. Saracco, "Symbiotic autonomous systems" (White Paper II, IEEE, 2018). https://www.diva-portal.org/smash/get/diva2:1260812/FULLTEXT02.pdf. Accessed 6 July 2020.
18. J. Sifakis, Autonomous systems an architectural characterization" in *Models, Languages, and Tools for Concurrent and Distributed Programming*, M. Boreale, F. Corradini, M. Loreti, R. Pugliese, Eds. (Springer, 2019) pp. 388–410.
19. J. Sifakis, Rigorous system design. *Found. Trends Electron. Des. Autom.* **6**, 293–362 (2013).
20. D. Harel, A. Pnueli, *On the Development of Reactive Systems* (NATO ASI Series, Springer, New York, 1985), vol. F-13.
21. S. Boschert *et al*., "Symbiotic autonomous systems," T. Cavrak, Ed. (White Paper III, IEEE, 2019) https://digitalreality.ieee.org/images/files/pdf/1SAS_WP3_Nov2019.pdf. Accessed 6 July 2020.
22. P. G. Neumann, How might we increase system trustworthiness? *Commun. ACM* **62**, 23–25 (2019).
23. ACM, ACM transactions on autonomous and adaptive systems (TAAS). https://dl.acm.org/journal/taas. Accessed 6 July 2020.
24. C. K. Chang, Situation analytics: A foundation for a new software engineering paradigm. *Computer* **49**, 24–33 (2016).
25. C. K. Chang, Situation analytics—at the dawn of a new software engineering paradigm. *Sci. China Inf. Sci.* **61**, 050101 (2018).
26. B. Douglas, Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM* **38**, 33–38 (1995).
27. W. Damm, E. Möhlmann, T. Peikenkamp, A. Rakow, "A formal semantics for traffic sequence charts" in *Principles of Modeling*, M. Lohstroh, P. Derler, M. Sirjani, Eds. (Springer, 2018), pp. 182–205.
28. N. G. Wassim, J. D. Smith, M. Yanagisawa, "Pre-crash scenario typology for crash avoidance research" (Tech. Rep. DOT HS 810 767, National Highway Traffic Safety Administration, 2007).
29. S. Kato *et al*., An open approach to autonomous vehicles. *IEEE Micro* **35**, 60–68 (2015).
30. H. Watanabe, L. Tobisch, J. Rost, J. Wallner, G. Prokop, "Scenario mining for development of predictive safety functions" in *2019 IEEE International Conference of Vehicular Electronics and Safety (ICVES)* (IEEE, 2019), pp. 1–7.
31. D. Lo, S. Maoz, S.-C. Khoo, "Mining modal scenario-based specifications from execution traces of reactive systems" in Proceedings of the *22*nd IEEE/ACM *International Conference* on Automated *Software Engineering* (ACM, 2007), pp. 465–468.
32. C. Bertero, M. Roy, C. Sauvanaud, G. Trédan, "Experience report: Log mining using natural language processing and application to anomaly detection" in *28th International Symposium on Software Reliability Engineering (ISSRE)* (IEEE, 2017).
33. Z. Tahir, R. Alexander, "Coverage based testing for V&V and safety assurance of self-driving autonomous vehicle: A systematic literature review" in *The Second IEEE International Conference on Artificial Intelligence Testing* (IEEE, Oxford, UK, 2020).
34. Waymo, JOURNEY. https://waymo.com/journey/. Accessed 6 July 2020.
35. Venturebeat, Uber's 250 autonomous cars have driven "millions" of miles and transported "tens of thousands" of passengers. https://venturebeat.com/2019/04/11/ubers-250-autonomous-cars-have-driven-millions-of-miles-and-transported-tens-of-thousands-of-passengers/. Accessed 6 July 2020.
36. T. F. Kone, E. Bonjour, E. Levrat, F. Mayer, S. Géronimi, "Safety demonstration of autonomous vehicles: A review and future research questions" in *Proceedings of the Tenth International Conference on Complex Systems Design & Management, CSD&M Paris 2019*, G. A. Boy, A. Guegan, D. Krob, V. Vion, Eds. (Springer, 2019), pp. 176–188.
37. P. Bouyer *et al*., Reasoning about quality and fuzziness of strategic behaviours. arXiv:1905.11537 (27 May 2019).
38. I. Goodfellow *et al*., "Generative adversarial nets" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger, Eds. (Curran Associates, Inc., 2014), pp. 2672–2680.
39. G. Katz, C. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks" in *International Conference on Computer Aided Verification* (Springer, 2017), pp. 97–117.
40. D. Harel, A. Marron, A. Rosenfeld, M. Vardi, G. Weiss, "Labor division with movable walls: Composing executable specifications with machine learning and search (blue sky idea)" in Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI Press, 2019), vol. 33, pp. 9770–9774.
41. S. Shalev-Shwartz, S. Shammah, A. Shashua, On a formal model of safe and scalable self-driving cars. arXiv:1708.06374 (21 August 2017).
42. J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, J. Wu, The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. arXiv:1904.12584 (26 April 2019).