Exponential Contingency Explosion: Implications for Artificial General Intelligence

Samuel Haug, Robert J. Marks, II, *Life Fellow, IEEE*, and William A. Dembski, *Senior Member, IEEE*

***Abstract*—The failure of complex artificial intelligence (AI) systems seems ubiquitous. To provide a model to describe these shortcomings, we define complexity in terms of a system's sensors and the number of environments or situations in which it performs. The complexity is not looked at in terms of the difficulty of design, but in the final performance of the system as a function of the sensor and environmental count. As the complexity of AI, or any system, increases linearly the contingencies increase exponentially and the number of possible design performances increases as a compound exponential. In this worst case scenario, the exponential increase in contingencies makes the assessment of all contingencies difficult and eventually impossible. As the contingencies grow large, unexpected and undesirable contingencies are all expected to increase in number. This, the worst case scenario, is highly connected, or conjunctive. Contingencies grow linearly with respect to complexity for systems loosely connected, or disjunctive. Mitigation of unexpected outcomes in either case can be accomplished using tools such as design expertise and iterative redesign informed by intelligent testing.**

***Index Terms*—Artificial intelligence, complexity theory, machine intelligence, product safety engineering, robustness, self-driving cars, system design, system testing.**

I. Introduction

THE MORE complex a system, the greater number of performance contingencies. This growth is a concern requiring serious consideration in the pursuit of artificial general intelligence (AGI) [2], [29], [39]. The analysis herein does not address the goals of AGI research, but rather the difficulty of achieving them. Any effort at achieving AGI will, by necessity, be complex. Our analysis is applicable to all complex systems including complex artificial intelligence (AI).

After giving examples of unanticipated contingencies in complex systems, our analysis shows that a linear increase in system complexity can give an exponential explosion of contingencies. Methods to mitigate away from introduction of unexpected contingencies are then presented.

A. Examples of the Unforeseen

Undesirable and unexpected contingencies have already been manifest in the deployment of AI systems. Here are some examples of unexpected contingencies from complex systems ranking from the simply curious to the very serious.

1) During IBM Watson's ultimate win over two human contestants in the quiz show Jeopardy, a curious thing happened. A query under the category NAME THE DECADE was "The first modern crossword is published and Oreo cookies are introduced." A human, Ken Jennings, was first to hit the buzzer and responded "What are the 20's?" The quizmaster, Alex Trebek, ruled the response incorrect. IBM Watson then buzzed in and gave the same exact response. "What are the 20's?" Alex Trebek responded "No. Ken said that." The programmers of IBM Watson had made no provision for Watson to respond with an answer that had already been declared incorrect [45], [48]. Watson's duplicate response was an unintended contingency of the IBM software.[1]
2) In 1997 IBM's Deep Blue beat world champion Gary Kasparov at chess. One of Deep Blue's moves was particularly curious. The unexpected move psychologically threw Kasparov off his game and he lost. Of the move, one chess expert said [19]:

   > "It was an incredibly refined move, of defending while ahead to cut out any hint of countermoves."

   This comment is akin to a child's finger painting being lauded as inspirational modern art. Over a decade after the match, Murray Campbell, one of the three IBM computer scientists who designed Deep Blue, first confessed Deep Blue's move was a fluke—a bug in the code. Deep Blue was unable to select a move and simply picked one at random.

   > "Kasparov had concluded that the counterintuitive play must be a sign of superior intelligence, he had never considered that it was simply a bug."
3) A deep convolutional neural network was trained to detect wolves. After the trained neural network incorrectly classified a husky dog as a wolf, the programmers did some forensics and discovered there was undesirable bias in the training data. The pictures of wolves all contained snow. The picture of the misclassified dog also contained snow. In training, the neural network had learned the presence and absence of snow. The features

Manuscript received June 11, 2020; revised November 8, 2020; accepted January 20, 2021. Date of publication March 4, 2021; date of current version April 15, 2022. This work was supported in part by the Walter Bradley Center for Natural & Artificial Intelligence, Seattle, Washington. This article was recommended by Associate Editor L. Wang. *(Corresponding author: Samuel Haug.)*

Samuel Haug is with the Honors College, Baylor University, Waco, TX 76798 USA (e-mail: sam_haug@baylor.edu).

Robert J. Marks, II is with the Department of ECE, Baylor University, Waco, TX 76798 USA (e-mail: robert_marks@baylor.edu).

William A. Dembski is with the Evolutionary Informatics Laboratory, Aubrey, TX, USA (e-mail: billdembski@gmail.com).

Digital Object Identifier 10.1109/TSMC.2021.3056669

[1]Watson's mistake was anticipated as a possible AI flaw in 1974. In a major plot component on the television series *The Six Million Dollar Man* [47], Steve Austin, the six million dollar man, suspects a friend has been replaced by a robot. He asks a question to the suspected imposter and gets an answer. He then asks the same question, only to get the same answer. The exact repeated answer fails the Turing test and revealed the answer came from a machine and not a human.

of the animals were not considered in the classification problem [22].

4) An inconvenience for self-driving cars is false classification of objects like plastic bags. A stationary plastic bag can be categorized as a large rock [27] while a wind-blown plastic bag can be mistaken for a deer [37]. These are unintended contingencies of the self-driving car's software.
5) A more serious problem with self-driving cars is fatalities. In 2018, an Uber self-driving car in Tempe, Arizona struck and killed pedestrian Elaine Herzberg [36]. The incident was an unintended contingency and preventable. Steven Shladover, a UC Berkeley research engineer, noted "I think the sensors on the vehicles should have seen the pedestrian well in advance." The death was a tragic example of the an unintended contingency of a complex AI system. Unintended contingencies remain a major obstacle in the development of general (level 5) self-driving cars. Some developers, believing the problem in insurmountable, have given up [14].
6) During the height of the cold war, the Soviets deployed a satellite early warning system called Oko to watch for incoming missiles fired from the United States. On September 26, 1983, Oko detected incoming missiles. At a military base outside of Moscow, sirens blared and the Soviet brass was told by Oko to launch a thermonuclear counterstrike. Doing so would result in millions being killed. The officer in charge, Lieutenant Colonel Stanislav Petrov, felt something was fishy. After informing his superiors of his hunch Oko was not operating correctly, Petrov did not obey the Oko order. Upon further investigation, Oko was found to have mistakenly interpreted the sun reflected off of clouds [45]. There was no U.S. missile attack. Petrov's skepticism of Oko's alarm may have saved millions of lives.

These examples of unintended contingencies deal with systems of broad complexity. Narrow AI systems are typically more error free. Examples of a narrow AI system are anti-radiation missiles like Israel's Harpy. The missile is launched and flies about (loiters) over a predefined kill zone. The missile can operate autonomously without human oversight. If fuel gets low, the missile returns home. Alternately, if illuminated by radar, the anti-radiation missile zeros-in on the location of radar. The missile follows the radar beam back to its source and destroys the radar installation [35]. Whether or not one agrees with the mission of such a system, the anti-radiation missile is an example of relatively narrow AI that has historically worked without flaw. There are few if any unforeseen contingencies in anti-radiation missiles that will distract from its duties.

We revisit this list of unintended contingencies in Section IV-A where the cause of each of these events is categorized in terms of knowability.

B. Measuring Complexity

Complexity can be related to the number of sensors used in a system and the number of environments or situations in which the system must operate. We first analyze tightly coupled systems where sensor readings are combined conjunctively. For tightly coupled, or conjunctive systems, an increase of either parameter results in an exponential increase in contingencies. The tally of all possible design results grows as a compound exponential, like e^{e^x}, with respect to the number of system sensors and the number of environments or situations in which the system operates. The contingencies of disjunctive [17], [50], or loosely coupled systems, grows linearly with respect to complexity, (see Section VI-D). The number of possible designs grows exponentially. Hybrid systems composed of a disjunctive collection of small conjunctive systems lies in between in terms of contingency count and the total number of possible designs.

II. Background

An early effort listing of measures of system complexity was offered by Lloyd [30] who categorized complexity into three categories: 1) degree of organization and difficulties of; 2) creation; and 3) description. Carlson and Doyle [6] introduced the idea of highly optimized tolerance (HOT). HOT illustrates the tendency of highly optimized complex systems to perform their tasks well, but to be sensitive to perturbations in the form of design flaws and unexpected contingencies. Complexity has been defined as a function of system's degrees of freedom [20] and as a measure of system description [9]. Other work on the design of complex systems has concentrated on the difficulty of constructing a system [16], [49].

Our analysis approaches the problem differently. Complexity is defined by the number of system sensors and the number of environments or situations in which the system operates. We are interested neither in the difficulty of the design of a system nor the degree of creativity required but rather only in the final design. There is no consideration given to potential failures within the system. All components are assumed to operate with no flaws. With no apriori assumptions, how can the design deviate unacceptably from its desired function?

Such analysis is in concert with the no free lunch theorems (NFLTs) [10], [51] where no design expertise or, equivalently, no bias is assumed in a search process. In such a case, one search algorithm is shown to be no better on average than another [33]. In 1980, Mitchell seems to have first recognized the need for bias in search [38]. More recently the property, dubbed *conservation of information* [11]–[13], [33], has been further popularized by Schaffer [46] and Wolpert and MacReady [51].

Our analysis builds on that of Ho *et al.* [23] who showed that the analysis akin to the NFLT's could be applied to analyze certain complexity and security aspects of system design. They did so using the *fundamental matrix* similar in structure to the matrix used in Wolpert and MacReady's proof of the NFLT. We build on the insightful work of Ho *et al.* to show the number of possible designs of a conjunctive system increases as a compound exponential with respect to a linear increase in complexity. The number of contingencies increases exponentially with respect to complexity. Unexpected contingencies can be mitigated by the design expertise of the system designer and repeated design using results from intelligent

TABLE I
DESIGN PERFORMANCE MATRIX FOR WASHER WITH TWO SENSORS

$X \downarrow \vert Y \rightarrow$	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
$x_1 = (0,0)$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$x_2 = (0,1)$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$x_3 = (1,0)$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$x_4 = (1,1)$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

testing. The increase in contingencies can be so great, however, that consideration and testing of all cases becomes impossible.

III. DESIGN PERFORMANCE MATRIX

The *design performance matrix* enumerates all design possibilities of a system as a function of contingencies.

A. Example

The design performance matrix is best introduced by illustration. Consider the pedagogical example of a clothes washing machine that operates using only two sensors, namely, load weight and the dirtiness of the load. The load weight is easily measured. The dirtiness of the load is measured by the wash water's turbidity. This can be done by measuring the short distance attenuation of a submerged LED light source. To simplify the illustration, further assume there are only two readings from each sensor. The load will be either be "light" or "heavy" and the turbidity either "mild" or "dirty." We present these criteria in the spirit of fuzzy linguistic variables [4], [32], [52] recognizing that in the design, crisp numeric intervals must be defined for each case. Presentation using linguistic variables clarifies the presentation.

Binary variables are be assigned to each sensor

$$\text{weight} = (\text{light, heavy}) = (0, 1)$$
$$\text{turbidity} = (\text{mild, dirty}) = (0, 1).$$

The left most column in Table I lists all the sensor reading possibilities with the first entry denoting weight and the second turbidity. The contingency $x_3 = (1, 0)$ for example indicates the load is "heavy" and the turbidity "mild." Let A denote the set of sensors. We denote the set of all possible sensor readings in the left most column by X. Since there are $|A| = 2$ sensors, there are $|X| = 2^{|A|} = 4$ total possible sensor combinations where $|X|$ denotes the cardinality of the set X. The cardinality $|X|$ is the number of contingencies of the system.

The set of all possible design performance results is denoted by Y. Each design is denoted by a column in the design performance matrix. For the washer example in Table I, this set's cardinality is $|Y| = 16$.

A zero in the design performance matrix denotes failure to achieve pre-established performance criteria for the sensor reading combinations. An entry of one denotes success.

Consider the matrix row denoted by the sensor reading pair $x_2 = (0, 1)$. This entry means the load is "light" and the turbidity is "dirty." The design corresponding to the column y_7 in the design performance matrix is a one. This means that the design will work for light, dirty loads. The corresponding entry for design possibility y_8 is a zero meaning the design does not work for light, muddy loads. In the set of all possible designs, there is only one design that works for all possible sensor readings. That is, the design corresponding to y_{15} where the column has all ones.

TABLE II
MATRIX DESCRIPTION OF THE y_7 DESIGN IN TABLE I

turbidity $\rightarrow$	mild (0)	dirty (1)
light load (0)	0	1
heavy load (1)	1	1

Each column of the design performance matrix can itself be represented as a matrix. The column under y_7 in Table I, for example, can be written as shown in Table II. Table I is therefore seen to be a two-dimensional representation of a three-dimensional tensor [34].

There is not necessarily a single design corresponding to, say, y_7 in Table I. A number of designs can give the performance of the ones and zeros in the y_7 column. Any washer design, however, will be described by one of the y's in Table I when tested. It follows that, even in the absence of all knowledge about a design, all of the design columns in the matrix should not be considered equally probable. (Equal probability columns are an assumption made in the NFLTs [12], [23], [51].) The goal of the design performance matrix, rather, is to list all *possible* designs in the absence of any and all knowledge about a design.

B. Scaling Generalization

The simple example of the washing machine can be generalized.

1) Increasing the Sensor Count: Let A denote the set of sensors

$$A = \{a_1, a_2, \ldots, a_{|A|}\}.$$

For the washing machine example just presented, $|A| = 2$. A third washer sensor might indicate whether the wash fabrics are "light" (0) or "heavy" (1). When added, the number of sensors is $|A| = 3$.

If there are $|A|$ sensors each of which is divided into the two categories of design success or failure, there are

$$|X| = 2^{|A|}$$

sensor reading combinations to consider. Then there are

$$|Y| = 2^{|X|} = 2^{2^{|A|}} \tag{1}$$

possible designs. For the washer example, $|A| = 2$, $|X| = 4$ and $|Y| = 16$. The increase scales rapidly. For $|A| = 20$ sensors, there are over a million contingencies ($|X| = 1048576$) and $|Y| = 6.7 \times 10^{315652}$ possible designs.[2]

2) Increasing the Number of Design Rankings: Let C denote the set rankings for each design. For the example in Table I, the elements in each cell in the design performance matrix are $C = \{0, 1\}$ with 0 corresponding to failure and 1 to success. Therefore $|C| = 2$. The classification of a design need not be limited to success or failure. Performance on, say, a scale of one to $|C| = 10$ corresponds to allowance of each cell in the design performance matrix to be an integer from 1

[2]When encountering large numbers of this sort, it is customary to point out that the number of atoms in the known universe is about 10^{80}.

TABLE III
MULTIOBJECTIVE OPTIMIZATION EXAMPLE FOR WASHER

turbidity →	not clean (0)	clean (1)
too slow (0)	0	1
sufficiently fast (1)	2	3

to 10. Like a judge ranking Olympic ice skating competitions, the output of the wash can be ranked on a scale of 1 to 10. The parameter $|C|$ is the number of ways a design can be judged. Instead of the design performance matrix corresponding to a count in base 2, the matrix is now composed on a count in base 10. As a function of base $|C|$, the number of possible designs becomes

$$|Y| = |C|^{|X|} = |C|^{2^{|A|}}. \tag{2}$$

Equation (1) is a special case for $|C| = 2$.

There is a second useful way that the dimension of C is increased. The design ranking can be used in multiobjective optimization. The washing machine's binary ranking in performance is determined by the cleanliness of the clothes. "Clean" = 1 and "not clean" = 0. A second criteria can be the time required to complete the wash. The following variables are assigned: "sufficiently fast" = 1 and "too slow" = 0. The possible outcomes can be assigned numbers as illustrated in Table III. There are $|C| = 4$ possible outcomes. An outcome of 2 in the matrix means that the clothes were not clean but the washing speed was sufficiently fast.

3) Increasing Sensor Reading Partitions: Finally, there can be more than two partitions for each sensor. For the washer example, the weight classifications "light" and "heavy" can be extended to the performance criteria "light," "medium," "heavy," and "very heavy." Let S denote the set of the sensor partitions

$$S = \left\{s_1, s_2, s_3, \ldots, s_{|A|}\right\}.$$

If sensor a_2 corresponds to the weight of the wash, then from the running example, $s_2 = 4$ when the additional performance criteria are added. The X column in the design performance matrix is no longer restricted to binary and can contain non binary elements like $x_3 = (3, 0)$. The number of sensor reading possibilities, equal to the number of contingencies, is now

$$|X| = \prod_{j=1}^{|A|} s_j. \tag{3}$$

If all sensor partitions are equal ($s_j = s\ \forall j$), this simplifies to

$$|X| = s^{|A|}. \tag{4}$$

In general, the number of design possibilities follows as:

$$|Y| = |C|^{|X|} = |C|^{\prod_{j=1}^{|A|} s_j}. \tag{5}$$

The number of design possibilities when $s_j = s$ follows from (4) as:

$$|Y| = |C|^{|X|} = |C|^{s^{|A|}}. \tag{6}$$

Equation (2) is a special case for $s = 2$. The possible designs therefore increases as a compound exponential with respect to the number of sensors. Specifically

$$\log|Y| = s^{|A|} \log|C|. \tag{7}$$

a) Images: A simple example with an enormous number of sensors is images. A small $|A| = 100 \times 100$ image has ten thousand pixels. Each pixel has 256 gray levels times 3 color choices, e.g., RGB. That is, $s = 256 \times 3 = 768$. The number of test scenarios, from (4), is then

$$|X| = 768^{10000} = 10^{28854}. \tag{8}$$

If $C = \{0, 1\}$ for success and failure, we have, from (6)

$$|Y| = 2^{768^{10^4}} = 10^{10^{28853}}$$

which is an unimaginably large number.

IV. ENVIRONMENTAL AND SITUATION AUGMENTATION

Environmental and situational parameters play a role in system performance. A clothes washer might perform differently if the ambient room temperature is "cold" (0), "medium" (1), or "hot" (2). We can add environmental effects to the design performance matrix. The sensor vector can be augmented from $x_1 = (0, 1)$ to $x_1 = (0, 1; 2)$ meaning that the washer is running in a "hot" environment (with a "light" load and "dirty" water). The environmental parameters are entered to the right of the semicolon.

In our use of the term, variations in situations are considered environmental changes. The choice of detergents in the washer example might effect the design performance. Each detergent or detergent type could be another environmental consideration.

Let E equal the set of environments in which the design operates

$$E = \left\{e_1, e_2, e_3, \ldots, e_{|E|}\right\}.$$

Let the set of environment partitions be

$$T = \left\{t_1, t_2, t_3, \ldots, t_{|E|}\right\}.$$

If, for example, the third environmental parameter is e_3 = ambient temperature, and the temperature is ranked "cold," "medium" and "hot," then $t_3 = 3$ partitions.

The X matrix here is augmented from its original definition to now include environments. Specifically

$$X = \{A, E\} = \left\{a_1, \ldots, a_{|A|}, e_1, \ldots, e_{|E|}\right\}.$$

If there are sensors a_1, a_2 and a_3 ($|A| = 3$) and environments e_1 and e_2 ($|E| = 2$), then the sensor/environmental count is five: $x_n = (s_1, s_2, s_3; t_1, t_2)$. The number of contingencies in general is therefore

$$|X| = \prod_{n=1}^{|E|} t_n \prod_{j=1}^{|A|} s_j. \tag{9}$$

The number of possible designs explodes to

$$|Y| = |C|^{\left(\prod_{n=1}^{|E|} t_n \prod_{j=1}^{|A|} s_j\right)}. \tag{10}$$

If the number of partitions for sensor readings are all the same ($s_j = s\ \forall j$) and the number of partitions for all environments is the same ($t_n = t\ \forall n$), then (9) becomes

$$|X| = t^{|E|} s^{|A|} \tag{11}$$

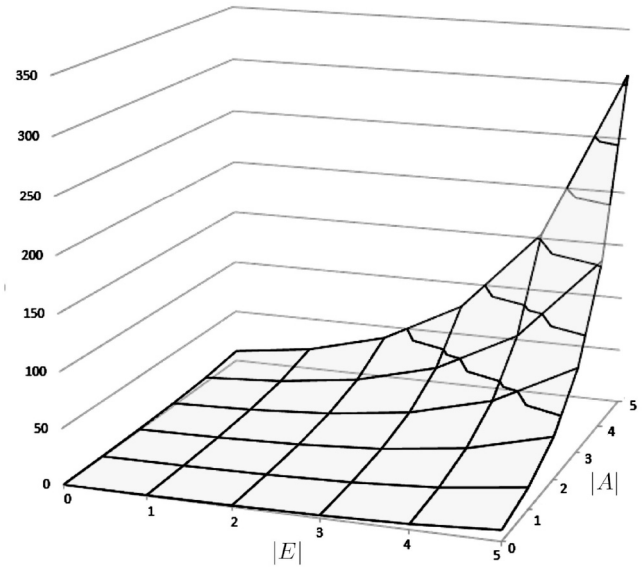


Fig. 1. Plot of $\log|Y|$ in (13) for $t = s = 2$. The log is base 10 so a value of 150 is interpreted here as $|Y| = 10^{150}$.

TABLE IV
SETS USED IN THE DESIGN PERFORMANCE MATRIX. PRIOR TO EACH ENTRY, READ "THE SET OF ⋯"

X = contingencies = all possible sensor and environment/situation combinations
Y = possible design performances
A = all sensors
C = rankings for each design performance scenerio
S = sensor reading partitions
E = all environments/situations
T = environmental/situational type partitions

and (10) simplifies to

$$|Y| = |C|^{\left(t^{|E|}s^{|A|}\right)} \tag{12}$$

or, equivalently

$$\log|Y| = t^{|E|}s^{|A|}\log C. \tag{13}$$

A plot is shown in Fig. 1 for $t = s = 2$. For $|E| = 3$ and $|A| = C = 2$, the number of design possibilities is $|Y| = 4294967296$.

For convenience, a list of all the sets used thus far is in Table IV.

V. WAYS TO FAIL

For a simple binary classification of success and failure in the design performance matrix ($|C| = 2$), Aristotle famously said:

> "It is possible to fail in many ways ⋯ while to succeed is possible only in one way."

This is evident in Table I where y_{15} is the only across-the-board success. All other design possibilities fail in one or more instances. Optimality, however, is typically not sought in a design. Rather, a list of specifications are made *a priori* to the design. Any design meeting these criteria is acceptable. From the perspective of the design performance matrix, many matrix columns can be claimed to be a successful design.

Failure of a design may require expanding to allow for different failure modes. The failure of a design can be relatively inconsequential or can be a significant undesirable consequent. The repeated failure of Alexa to respond to an oral command to play the 1958 pop classic *Get a Job* by The Silhouettes is annoying but inconsequential. The examples listed in the Introduction illustrate more substantial failures. In the design performance matrix a significantly undesirable consequent can be cataloged by replacing a 0 with, say, an 0*. By bifurcating the former 0s into 0s and 0*, we are in essence going into a ternary rather than binary characterization. But the contingency count does not change.

The examples listed in Introduction are all of undesirable consequences of system design and would be labeled 0* in the design performance matrix. A good design will have no 0*s in the matrix. But the existence of 0*s is unknown unless there is a revealing test. As the sensor count and environments increase, the number of undesirable consequences, i.e., the number of 0*s in the design performance matrix, can be expected to commensurately increase.

A. Knowability

The challenge of undesired consequences is their escape of notice by the system designer. The problem is well stated by Donald Rumsfeld who served as the United State's Secretary of Defense under Presidents Ford and George W. Bush [40]:

> "As we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we do not know we do not know."

Here, is a listing of the Rumsfeld categories as they relate to the design performance matrix.

1) The *known knowns* are the entries in design performance matrix about the known operation of the current design.
2) The *known unknowns* are elements in the design performance matrix that have yet to be determined by, for example, testing. There is no number entered into the design cell because the performance at this operating point is unknown.
3) The most troublesome design performances are the *unknown unknowns*. These are environments not included in the matrix that informed design expertise did not expect. In such cases, the design performance matrix is not large enough. More rows and columns need to be added. If the failure is result of an unanticipated situation, a new environmental variable is added to the set of contingencies and (11) will multiplicatively increase. That is, if an environmental variable is added, then

$$\begin{aligned}|X|_{\text{new}} &= t^{|E|+1}s^{|A|}\\ &= t\,|X|_{\text{old}}.\end{aligned} \tag{14}$$

Correspondingly, the number of possible designs increase by an exponential factor. From (12)

$$\begin{aligned}|Y|_{\text{new}} &= |C|^{\left(t^{|E|+1}s^{|A|}\right)}\\ &= |Y|_{\text{old}}^{t}.\end{aligned} \tag{15}$$

4) We define *unknown knowns*, not on Rumsfeld's list, as *overlooked knowns*. These are performance attributes that, once observed, are obvious and might prompt the designer to facepalm in embarrassment. Like unknown unknowns, these cases have been omitted from the environments in the design performance matrix. The omission is due to oversight rather that ignorance. The increases in (14) and (15) apply. Like unknown unknowns, these cases have been omitted from the environments in the design performance matrix. The omission is due to oversight rather that ignorance.

The four Rumsfeld categories relate to the AI failures listed in the Introduction which we now revisit applying the design performance matrix model.

1) In the Introduction deals with IBM Watson repeating a mistake previously ruled incorrect by the Jeopardy quizmaster. This is an example of an obvious missed environmental (situational) parameter of the designers. On the Rumsfeld list, it was an overlooked known. The occurrence must have embarrassed the designers of IBM Watson.
2) From the perspective of the overall chess game, the software mistake made by Deep Blue serendipitously helped the computer win. The win, though, was largely due to the psychological impact on Kasparov and not to any computer software. From the perspective of Deep Blue's performance, the move was a mistake due to a known unknown. All of the environmental (situational) cases had not been considered resulting in Deep Blue rolling the dice and choosing a move at random.
3) Convolutional neural networks can famously perform image classification by feeding pixel values directly into the classifier. The number of pixel inputs, each corresponding to a sensor, is enormous. For the conventional layered perceptron, the curse of dimensionality normally precludes using image pixels directly. Even though the number of required trainable weights in a convolutional neural network is reduced, the number of input pixels (sensors) is not. Testing all possible images corresponding, say, to a partition of 256 gray levels times three RGB levels for each pixel, is obviously prohibitive. Those that are not tested are known unknowns corresponding to nonentries in the design performance matrix. For the example of the dog mistaken as a wolf by a deep convolutional neural network because of snow in the background [3) in the Introduction], the mistake was discovered by testing.
4) Self driving cars mistaking plastic bags as rocks and deer is best categorized as an overlooked known. Deep design expertise in driving should have anticipated driving hazards like highway tread detached from tires of 18 wheelers (called *road gators* by truckers), wind blown plastic bags, and debris blown from the top of dump trucks in front of cars. For plastic bags, the experts writing the software did not anticipate such an occurrence. They should have.
5) According to the National Transportation Safety Board (NTSB), the failure of the lethal Uber self-driving car discussed in 5) in the Introduction was due to "'a fusion' of three sensor systems: a) radar; b) lidar; and c) a camera designed to detect an object and determine its trajectory." "The system design did not include a consideration for jaywalking pedestrians," the NTSB report said [18]. The self-driving car's deadly mistake is best categorized as an overlooked known. A driving expert would know that cars interact with pedestrians.
6) The Soviet Oko system that falsely announced American missiles had been launched could have resulted in the loss of millions of lives. This unanticipated unknown in the system design later became evident after further investigation.

VI. Design Performance Mitigation

The explosion of design possibilities in (10) and (13) assume total lack of knowledge concerning a design under inspection except for sensor count and environmental factors. If true in reality, the design of anything complex looks to be nearly impossible. But in practice the threat of possible contingencies is lesser by a number of factors. Foremost are inherent design limitations, domain expertise and testing. Approaching design from a disjunctive perspective can help the design process.

Even if the number of designs is reduced, the number of contingencies to label, $|X|$ given in (9) and (11), remains the same. The explosion remains exponential. If all four parameters in (9) are set to 5, there are still nearly 10 million contingencies to consider.

A. External Design Limitations

The probability of some unconsidered contingencies might be so small as to escape consideration. Some designs in the design performance matrix might not be possible for various reasons. In the design of a road vehicle, for example, the attributes of "safe" and "inexpensive" might not be possible to achieve simultaneously. Safe vehicles like Humvees are not cheap and cheap vehicles like scooters are not safe. It follows that a perfect design corresponding to all 1s in a column of the design performance matrix might not be achievable. Such Pareto tradeoffs [3], [7] are characteristic of multiobjective design. Knowledge of the impossibility of some designs due to factors, such as physics, financial, and social constraints, reduces the number of contingencies requiring testing. The compounded exponential growth in Fig. 1 may diminish, but the number of test scenarios, $|X|$, still remains exponential.

B. Testing

Design is inherently an iterative process [32], [49]. A prototype is built and tested. Deficiencies and mistakes occur from which the designer learns and improves the design. IBM Watson's mistake of repeating a wrong answer, for example, appears easily corrected.

Testing reveals unknown unknowns and overlooked contingencies. Once discovered, the design performance matrix may require augmenting to include one or more new environmental parameters.

C. Domain Expertise

Domain expertise is of singular importance in successful design of a complex system. Testing all potential contingencies in some sequential order is clearly not possible because of the large count in even moderately complex designs. A designer experienced in the field can better place the initial prototype in the ballpark closer to a successful design.

In the washer example, an expert might know the worst case detergent to use in the washing tests. Testing of other detergents would then be unnecessary.

Likewise, consider the $|X| = 768^{10000} = 10^{28854}$ images of size 100×100 in (8). This image set includes a lot of nonsense images like the instantiations of all possible combinations of random noise. These images are unlikely to be encountered. In this sense, the figure is overstated. A more reasonable set of images might be those compressible into, say, a lossless png file with size below some threshold.

Domain experience is also necessary in formatting informed testing scenarios. Formula 409, the spray cleaner sold by the Clorox company, was perfected after 409 attempts to achieve a design criterion. Hence, the name. A similar example is the petroleum-based lubricant and protectant, WD-40, invented in 1953 by Norman B. Larsen. We know the design was iterative because WD-40 stands for "water displacement, formulation successful in 40th attempt." A high school student just completing an introductory course in chemistry would require far more that the 40 iterations required by industrial chemist Norman B. Larsen. We would instead be using something named WD-1024 [33].

D. Disjunctive Design

The design performance matrix allows conjunctive high connectivity among components. The lethal Uber self-driving car killed because of a clash between three sensors: 1) "radar; 2) lidar; and 3) a camera." The clash contributed to the faulty AI decision process. Disjunctive system design [17], [50] has loose or no connection among components.

Loosely connected systems of smaller complexity are more robust and less subject to unintended contingencies than larger highly interconnected systems.

1) Disjoint System Examples: Examples of a disjunctive approach include disjoint functionality and redundancy. Disjoint functionality separates different system operations into silos. Designing and testing each silo independently is less complex than an overall test. Once established, however, the question of performance of the combination of the silos remains to be examined.

a) Disjoint functionality: Examples of disjoint functionality disjunctive design include the *Aegis Combat System* and Combs fuzzy control.

1) *Aegis:* The *Aegis Combat System* is a powerful and versatile weapon of the U.S. Navy. Scharre [45] gives a useful assessment of the system. "Aegis is a weapon system of staggering complexity." Aegis's composition as a number of individual subsystems of radar, missile and other capabilities allows for robust operation. Aegis *doctrines* select from the smörgåsbord of resources to accomplish a mission. Captain Pete Galluch, Commander of the Aegis Training and Readiness Center, says "You can mix and match [Aegis resouces]. It is a very flexible system," some doctrines are predetermined or can be composed by an operator. "We can do all doctrine statements, some with a push of a button, or bring them up individually." The *Aegis Combat System* is a loosely connected aggregation of simpler systems. The composite "system is of staggering complexity" yet boasts a history of reliable performance.
2) *Combs Control:* In fuzzy control, Combs and Andrews [8] proposed use of a disjunctive configuration to mitigate the rule explosion characteristic of Mamdani control [31], [44], [50]. Combs control makes use of the fact that many sensor or sensor combinations can achieve the same goal. A car can turn right by "turning the steering wheel right AND lightly breaking on the car's right tires AND slightly accelerating the car's left tires." Combining these resources into various control rules is characteristic of the conjunctive Mamdani approach. Combs would alter this to "turning the steering wheel right OR lightly breaking on the car's right tires OR slightly accelerating the car's left tires." The Combs approach allows the flexibility of use of one or more of the resources to solve the problem. Mamdani requires all resources be used in each rule. The use of AND and OR here clearly differentiates between a conjunctive versus a disjunctive design approach. For more detailed explanation of Combs control and its properties, see Ewert *et al.* [17].
3) *Redundancy:* Redundancy is often used to boost a system's reliability [28]. Since we assume all system components will operate without hardware failure, reliability in the conventional sense is not considered in our model as a contingency. Final performance is, rather, the focus. Swarm intelligence [5] is an example of redundant disjunctive design. Using simple rules, swarms can demonstrate gestalt emergent behavior [21]. Swarms can adapt [41], [42], e.g., worker ants taking on army ant properties when their swarm is attacked [5]. Swarms need not be mobile. Individual cells in sections of the human lung operate disjunctively to allow the emergent behavior of breathing. Swarm intelligence has found numerous applications [5], including search [26], optimization [15], and telecommunications [24], [25]. Offensive autonomous drone swarms are perhaps the most chilling application of swarm intelligence [35], [45]. Even when highly decimated, redundancy makes swarms robust in completing their mission [43].

2) Reduction in Contingencies for Disjunctive Design: Here is the analysis of the contingency count for a purely disjunctive system.

Let a system be composed of $|A|$ sensors and assume the influence of the action prescribed by one sensor is independent from the actions of the others. An example of this disjunctive relationship among $|A| = 3$ sensors is illustrated in the design

TABLE V
DISJUNCTIVE PERFORMANCE DESIGN MATRIX FOR $|A|$ = THREE BINARY SENSORS

$X \downarrow \vert Y \rightarrow$	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	$\cdots$	y_{62}	y_{63}
$x_1 = (0, -, -)$	0	0	0	0	0	0	0	0	0	0	$\cdots$	1	1
$x_2 = (1, -, -)$	0	0	0	0	0	0	0	0	0	0	$\cdots$	1	1
$x_3 = (-, 0, -)$	0	0	0	0	0	0	0	0	1	1	$\cdots$	1	1
$x_4 = (-, 1, -)$	0	0	0	0	1	1	1	1	0	0	$\cdots$	1	1
$x_5 = (-, -, 0)$	0	0	1	1	0	0	1	1	0	0	$\cdots$	1	1
$x_6 = (-, -, 1)$	0	1	0	1	0	1	0	1	0	1	$\cdots$	0	1

performance matrix shown in Table V. Assume each sensor has $s = 2$ partitions. Because the system is purely disjunctive, each sensor can be assessed separately. The number of contingencies is now reduced to $|X| = 6$. In general, the number of contingencies for a purely disjunctive system is

$$|X| = \sum_{j=1}^{|A|} s_j. \tag{16}$$

When the partition count is the same for all sensors ($s_j = s \ \forall j$), then (16) becomes

$$|X| = s \ |A|. \tag{17}$$

Compare with (4). Instead of the exponential growth of contingencies with respect to the number of sensors, the growth is linear.

There are $|C| = 2$ rankings for each sensor in the example in Table V. A one in the design performance matrix means that the action from a sensor is successfully contributing to a specific planned mission. A zero means it is not. The conjunctive design analysis in (2) shows the number of possible designs increasing as a compound exponential. Using (17), the number of possible designs is seen to increase only exponentially for the disjunctive case. Specifically

$$|Y| = |C|^{|X|} = |C|^{\sum_{j=1}^{|A|} s_j} = \prod_{j=1}^{|A|} |C|^{|A| s_j} \tag{18}$$

and, if $s_j = s \ \forall j$, then

$$|Y| = |C|^{|X|} = |C|^{s|A|}. \tag{19}$$

The treatment of environments/situations does not follow in disjunctive design. It makes no sense to *a priori* assess environments/situations using disjunctive analysis in the design. We are constrained to conjunctive combinations. With reference to (9), the total number of contingencies for a disjunctive design is

$$|X| = (s \ |A|) \prod_{n=1}^{|E|} t_n$$

or, if $t_n = t \ \forall n$

$$|X| = (s \ |A|) t^{|E|}.$$

So with disjunctive design, consideration of environments and situations remains basically conjunctive.

Alternately, there are cases where environments in which a system finds itself can be tested before use. The Aegis system can vet its current environment by safe testing. "Once an Aegis able ship arrives in a theater, the first thing the crew does is test the weapons doctrine to see if there is anything in the environment that might cause it to fire in peacetime $\cdots$ This is done safely by enabling a hardware-level cutout called the *Fire Inhibit System* $\cdots$" Commander Galluch summarizes: "there is no voltage that can be applied to light the wick and let the rocket fly out" [45]. Such testing helps avoid unknown unknowns. Testing system operation prior to system deployment, when possible, eliminates the exponential explosion of contingencies required for preplanned environments and situations in the design.

VII. CONCLUSION

Increasing complexity increases operational contingencies. We can reasonably expect the increase in the number of contingencies will contain unexpected contingencies. Depending on how the system is used, these unexpected contingencies can range from the annoying to the highly serious. The worse case of contingency growth is a highly connected, or conjunctive, system where the contingencies grow exponentially with respect to an increase in complexity. On the other end is the loosely connected disjunctive system where an increase in the number of sensors yields a linear increase in contingency count. Most system designs may lie between the two extremes of totally conjunctive and disjunctive. The avoidance of unexpected contingencies can be mitigated by domain expertise. Insightful testing can be used by the designer to refine the design. Design is inherently iterative.

Although the analysis is applicable to all system design, it is especially troubling for the design of AGI. By necessity, general AI requires high complexity which will eventually birth contingency counts difficult to address.

REFERENCES

[1] P. Arabshahi *et al.*, "Adaptive routing in wireless communication networks using swarm intelligence," in *Proc. 19th AIAA Int. Commun. Satellite Syst. Conf.*, Toulouse, France, Apr. 2001, pp. 1–9.

[2] M. Baroni *et al.*, "CommAI: Evaluating the first steps towards a useful general AI," 2017. [Online]. Available: arXiv:1701.08954.

[3] C. Baylis, R. J. Marks, and L. Cohen, *Pareto Optimization of Radar Receiver Low-Noise Amplifier Source Impedance for Low Noise and High Gain*. Cambridge, U.K.: Cambridge Univ. Press, Nov. 2015.

[4] J. C. Bezdek, "Fuzzy models—What are they, and why?" *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 1–6, Feb. 1993.

[5] E. Bonabeau, M. Dorigo, G. Theraulaz, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY, USA: Oxford Univ. Press, 1999.

[6] J. M. Carlson and J. Doyle, "Highly optimized tolerance: Robustness and design in complex systems," *Phys. Rev. Lett.*, vol. 84, no. 11, pp. 2529–2532, Mar. 2000.

[7] Y. Censor, "Pareto optimality in multiobjective problems," *Appl. Math. Optim.*, vol. 4, no. 1, pp. 41–59, 1977.

[8] W. E. Combs and J. E. Andrews, "Combinatorial rule explosion eliminated by a fuzzy rule configuration," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 1–11, Feb. 1988.

[9] M. Danishvar, A. Mousavi, and P. Broomhead, "EventiC: A real-time unbiased event-based learning technique for complex systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 5, pp. 1649–1662, May 2020.

[10] W. A. Dembski, *No Free Lunch: Why Specified Complexity Cannot Be Purchased Without Intelligence*. New York, NY, USA: Rowman Littlefield, 2006.

[11] W. A. Dembski and R. J. Marks, II, "Conservation of information in search: Measuring the cost of success," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, no. 5, pp. 1051–1061, Sep. 2009.

[12] W. A. Dembski and R. J. Marks, "Bernoulli's principle of insufficient reason and conservation of information in computer search," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2009, pp. 2647–2652.

[13] W. A. Dembski and R. J. Marks, II, "The search for a search: Measuring the information cost of higher level search," *J. Adv. Comput. Intell. Intell. Informat.*, vol. 14, no. 5, pp. 475–486, 2010.

[14] B. Dixon, *Star Self-Driving Truck Firm Shuts Down; AI Not Safe Enough Soon Enough*, Mind Matters News, Seattle, WA, USA, Mar. 2020.

[15] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[16] K. Dorst and N. Cross, "Creativity in the design process: Co-evolution of problem-solution," *Design Stud.*, vol. 22, no. 5, pp. 425–437, Sep. 2001.

[17] W. Ewert, R. J. Marks, B. B. Thompson, and A. Yu, "Evolutionary inversion of swarm emergence using disjunctive combs control," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1063–1076, Sep. 2013.

[18] R. Gonzales, *Feds Say Self-Driving Uber SUV Did Not Recognize Jaywalking Pedestrian in Fatal Crash*, Washington, DC, USA, NPR, Nov. 2019. [Online]. Available: npr.org

[19] K. Finley. (Sep. 2012). *Did a Computer Bug Help Deep Blue Beat Kasparov?*. [Online]. Available: Wired.com

[20] H. Gao, C. Ye, W. Lin, and J. Qiu, "Complex workpiece positioning system with nonrigid registration method for 6-DoFs automatic spray painting robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Mar. 31, 2020, doi: 10.1109/TSMC.2020.2980424.

[21] I. A. Gravagne and R. J. Marks, II, "Emergent Behaviors of Protector, Refugee, and Aggressor Swarm," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 37, no. 2, pp. 471–476, Apr. 2007.

[22] P. Haas. (Dec. 2017). *The Real Reason to Be Afraid of Artificial Intelligence, TEDxDirigo*. Accessed: Apr. 4, 2020. [Online]. Available: youtu.be/TRzBk_KuIaM

[23] Y.-C. Ho, Q.-C. Zhao, and D. L. Pepyne, "The no free lunch theorems: Complexity and security," *IEEE Trans. Autom. Control*, vol. 48, no. 5, pp. 783–793, May 2003.

[24] I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks, II, P. Arabshahi, and A. A. Gray, "Swarm intelligence for routing in communication networks," in *Proc. IEEE Global Telecommun. Conf. (Globecom)*, San Antonio, TX, USA, 2001, pp. 3613–3617.

[25] I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks, II, P. Arabshahi, and A. A. Gray "Adaptive-SDR: Adaptive swarm-based distributed routing," in *Proc. Int. Joint Conf. Neural Netw. World Congr. Comput. Intell.*, Honolulu, HI, USA, May 2002, pp. 2878–2883.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, 1995, pp. 1942–1948.

[27] L. Kong, M. K. Khan, F. Wu, G. Chen, and P. Zeng, "Millimeter-wave wireless communications for IoT-cloud supported autonomous vehicles: Overview, design, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 62–68, Jan. 2017.

[28] L. M. Leemis, *Reliability: Probabilistic Models and Statistical Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995.

[29] A. Lieto, M. Bhatt, A. Oltramari, and D. Vernon, "The role of cognitive architectures in general artificial intelligence," *Cogn. Syst. Res.*, vol. 48, pp. 1–3, May 2018.

[30] S. Lloyd, "Measures of complexity: A nonexhaustive list," *IEEE Control Syst. Mag.*, vol. 21, no. 4, pp. 7–8, Aug. 2001.

[31] E. H. Mamdani, "Advances in the linguistic synthesis of fuzzy controllers," *Int. J. Man-Mach. Stud.*, vol. 8, no. 6, pp. 669–678, 1976.

[32] R. J. Marks-II, Ed. and L. A. Zadeh, *Fuzzy Logic Technology and Applications*. New York, NY USA: IEEE Technol. Activities Board, 1994.

[33] R. J. Marks, W. A. Dembski, and W. Ewert, *Introduction to Evolutionary Informatics*. Singapore: World Sci., 2017.

[34] R. J. Marks, II, *Handbook of Fourier Analysis & Its Applications*. New York, NY, USA: Oxford Univ. Press, 2009.

[35] R. J. Marks, *The Case for Killer Robots: Why Americas Military Needs to Continue Development of Lethal AI*. Seattle, WA, USA: Discov. Inst. Press, 2020.

[36] A. Marshall, *The Uber Crash Won't Be the Last Shocking Self-Driving Death*, WIRED, Boone, IA, USA, Mar. 2018. [Online]. Available: https://www.wired.com/story/uber-self-driving-crash-explanation-lidar-sensors/

[37] N. McBride, "The ethics of driverless cars," *ACM SIGCAS Comput. Soc.*, vol. 45, no. 3, pp. 179–184, 2016.

[38] T. M. Mitchell, "The need for biases in learning generalizations," Dept. Comput. Sci., Rutgers Univ., New Brunswick, NJ, USA, Rep. CBM-TR-117, p. 59, 1980.

[39] V. C. Müller, "Risks of general artificial intelligence," *J. Exp. Theor. Artif. Intell.*, vol. 26, no. 3, pp. 297–301, 2014.

[40] R. Pawson, G. Wong, and L. Owen, "Known knowns, known unknowns, unknown unknowns: The predicament of evidence-based policy," *Amer. J. Eval.*, vol. 32, no. 4, pp. 518–546, 2011.

[41] J. H. Roach, W. Ewert, R. J. Marks, II, and B. B. Thompson "Unexpected emergent behaviors from elementary swarms," in *Proc. IEEE 45th Southeastern Symp. Syst. Theory (SSST)*, Waco, TX, USA, Mar. 2013, pp. 41–50.

[42] J. H. Roach, R. J. Marks, II, and B. B. Thompson "Tactical task allocation and resource management in non-stationary swarm dynamics," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Dallas, TX, USA, Aug. 2013, pp. 1–5.

[43] J. H. Roach, R. J. Marks, II, and B. B. Thompson, "Recovery from sensor failure in an evolving multiobjective swarm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 1, pp. 170–174, Jan. 2015.

[44] M. G. Sanchez-Torrubia, C. Torres-Blanc, and S. Krishnankutty, "Mamdani's fuzzy inference eMathTeacher: A tutorial for active learning," *WSEAS Trans. Comput.*, vol. 7, no. 5, pp. 363–374, 2008.

[45] P. Scharre, *Army of None: Autonomous Weapons and the Future of War*. New York, NY, USA: WW Norton Company, 2018.

[46] C. Schaffer, "A conservation law for generalization performance," in *Proc. 11th Int. Conf. Mach. Learn.*, 1994, pp. 259–265.

[47] *The Six Million Dollar Man, Season One, Episode 4, Day of the Robot*, IMDb, Seattle, WA, USA, 1974. [Online]. Available: https://www.imdb.com/title/tt0071054/

[48] G. Smith, *The AI Delusion*. Oxford, U.K.: Oxford Univ. Press, 2018.

[49] H. Takeda, P. Veerkamp, and H. Yoshikawa, "Modeling design process," *AI Mag.*, vol. 11, no. 4, p. 37, 1990.

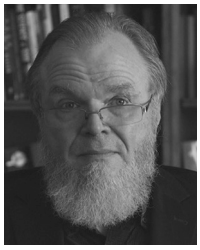
[50] J. J. Weinschenk, W. E. Combs, and R. J. Marks, II, "Avoidance of rule explosion by mapping fuzzy systems to a disjunctive rule configuration," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, St. Louis, MO, USA, May 2003, pp. 43–48.

[51] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[52] Z. Xu, "Group decision making with triangular fuzzy linguistic variables," in *Proc. Int. Conf. Intell. Data Eng. Autom. Learn.*, 2007, pp. 17–26..

Samuel Haug received the B.A. degree with Baylor University, Waco, TX, USA.

He is currently taking a gap year before attending medical school, during which he is participating in research with the Department of Engineering and Computer Science, Baylor University.



Robert J. Marks, II (Life Fellow, IEEE) is a Distinguished Professor of Engineering with the Department of Engineering and Computer Science, Baylor University, Waco, TX, USA, and the Director and a Senior Fellow of the Walter Bradley Center for Natural and Artificial Intelligence, Baylor University, where he is the Faculty Advisor for the Ratio Christi and American Scientific Affiliation student chapters. He has the authored/coauthored *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks* (MIT Press), *Handbook of Fourier Analysis and Its Applications* (Oxford University Press), *Introduction to Shannon Sampling and Interpolation Theory* (Springer-Verlag), and *Introduction to Evolutionary Informatics* (World Scientific).

William A. Dembski (Senior Member, IEEE) received the Master of Divinity in theology from the Princeton Theological Seminary, the Ph.D. in mathematics from the University of Chicago, and another Ph.D. in philosophy from the University of Illinois at Chicago.

He is a Writer, a Researcher, and a Entrepreneur. A Cross-Disciplinary Scholar with doctorates in mathematics and philosophy, he has contributed to the peer-reviewed literature in mathematics, engineering, philosophy, and theology. He has authored and edited over twenty books, many of them on aspects of the theory of evolutionary informatics, notably *The Design Inference: Eliminating Chance Through Small Probabilities* (Cambridge University Press, 1998). His main work these days focuses on developing educational technologies, especially data analytics tools for assessing influence in education. He has recently coauthored with Robert J. Marks a biography of Walter Bradley titled *For a Greater Purpose*. A baseball fan, he also coauthored (with Alex Thomas and Brian Vikander) the story of Steve "White Lightning" Dalkowski, reputed to be the fastest pitcher ever. This biography is titled *Dalko: The Untold Story of Baseball Fastest Pitcher* (Influence Publishers, 2020).