特集

On explainability in digital computing environments:
From the view of logical verification and proof assistants

Mitsuhiro Okada Keio University

1. Introduction

Societies and individual lives are surrounded by new digital and machine learning-AI environments. Many issues have been raised and discussed regarding the fairness, equity, explainability, transparency, and other digital ethics of these new digital and machine learning-AI environments. Practical solutions to these issues facing society will be important. On the one hand, there is a strong need for "basic research" involving understanding of various fundamental notions of digital ethics. On the other hand, it should be noted that some of the ethical issues of the digital environment, such as explainability of computational processes and fairness of information presentation have already been apparent with the emergence of the software-network society in the late 20th century. We believe that understanding the above basic notions that emerged in the pre-machine learning environments (namely, the software and communication network environments) will be useful in examining current machine learning and big data environments. As shown below, the information processing environment is divided into A-level (the manual process environment), B-level (the software and communication network environment), and C-level (the machine learning and big data environment). We will focus on such perspectives. Of course, there is a view that the C-level environment is a new environment that has never existed before, and that the issues of digital ethics that have emerged in this environment are a completely new set of issues. However, even if we take, for example, the fairness of the digital environment as an example, it is not only a problem specific to data science or AI models, but it also has a certain continuity with the Civil Rights Movement of the 20th century and related to traditional philosophical, ethical, and legal discussions. Clearly, the role of contemporary digital science and technology is central when addressing a set of contemporary issues in the digital environment at the C-level, but at the same time, fundamental understandings of the issues and debates inherited from the humanities and social sciences must also be fully taken into account. This article attempts to rethink the issue of "explainability in the digital environment," which has potentially already emerged in the B-level software and the communication network environment, from the perspective of logic, which has aspects of both the humanities and the mathematical sciences. This article is compiled based on the author's presentations at three recent French-Japanese international conferences, Explainability and Fairness in Digital Environments (March 27, 2025, Keio University), Towards the Fair Developments of Digital Environments, including OA and OS (March 24, 2025, Keio University), Education and AI (March 23, 2025, Keio University), and on the author's keynote talk at the CIPSH (International Council of Philosophy and Human Sciences) 75th Anniversary Conference, Beijing, September, 23, 2024.

2. Background

In this section, we discuss the explainability of software programs from the perspective of logic. The readers are reminded that logic has a humanities (and social sciences) aspect as well as a mathematical, scientific, and technical aspect. We aim to provide an example of cooperation between the humanities and scientific/technical perspectives. For example, issues of fairness, explainability, and transparency in the broad sense of digital ethics are not unique to current AI and big data environments. They include many issues that appeared in manual-based information processing environments even before the software and communication network environments entered society and individual lives. The author proceeds the discussion in this article with the following assumptions about the epochal changes in information processing in society.

Since the late 2010s, AI ethics and guidelines have been issued at the regional level (e.g., EU's Ethics guidelines for trustworthy AI, 2019) and at the level of AI-related societies, and it was particularly significant that in November 2021, UNESCO announced the Recommendation on the Ethics of Artificial Intelligence at the global level. It is important to note here that the November 2019 UNESCO World Logic Day was established [(1)] and the Proclamation Statement of UNESCO Director-General, Audrey Azoulay, on the establishment of a World Logic Day, made it clear that UNESCO World Logic Day is a day to recognize and celebrate the contributions of logic, but

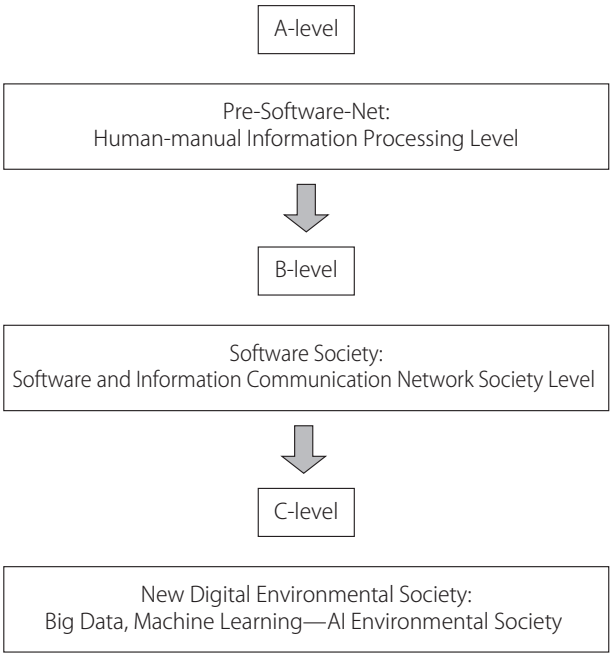

Figure 1: 3-level epochal changes of information processing environments

that this contribution of logic to humanity rightly includes the development of computers and AI as well as sciences, along with the human's ability of logical thinking, and the statement mentioned the guidelines for human-centralized AI was in course of preparation. Namely, in establishing this UNESCO World Logic Day, UNESCO has announced that it is preparing guidelines on ethics for human-centered AI. The Recommendation on the Ethics of Artificial Intelligence announced will be published in November 2022.[(2)]

3. Formal verification of program correctness, proof assistant systems and explainability with logical explanatory proofs

3.1 Preliminary

This section focuses on the issue of logically grasping the explainability issue, as an issue at the B-level environment, and an overview from a logical perspective of the extent to which this attempt at explaining is successful.

With the B-level software environments, when the source code is open, the programmers could detect how a result of a program execution came with the code. The programmer may be able to explain how the result came up to non-programmers, or to ordinary people. This is of course the case when a program is rather simple. If the program is small enough, the whole procedure of the program could be drawn as a flowchart or diagram on paper, which are manual-like tools. Note that an 8 set of manual(s) to instruct how to handle a procedure is a typical tool in the A-level environments. But, when one considers large-scale software, it is less easy to analyze programs involved, to explain, for example, how and why this or that result came up. For example, a software of a large scale nation-wide banking system or stock market system comes up with an alert (of system error) to shut down the nationwide system. The expert software engineer team often needs hours or days to try to detect it. Although this may be considered a rather rare case, one cannot always expect explanation of the result of software execution to the software development experts. In general, code analyses are not easy to prepare an explanation to the software users, especially when the purpose and specification of the program-software are not targeted to the explanations. Explanations of the execution results could be provided by the program-software side to some extent when the specification of the program-software is precisely described. A loan applicant who received a denial can see all the possible cases to reach a denial result by a precisely described program specification; it means the applicant's case is one of these denial cases. A specification could be completed, and even if the expert can detect how the execution process reached a denial with this case (which is not always easy), expressing the process to reach the denial as an explanation usually requires changing the procedural description of the execution process of the usual programming language into an explanation with declarative (natural) language.

The formal specification methodology is expected to be ideal for making a specification of a program-software precisely. To detect a program execution process either statically or dynamically, the logical formal verification method is promising. In particularly, we will refer to the type theoretic logical verification method, in which a program is identified with a logical (declarative) proof. We will also refer to the formal verification method with Hoare logic (and dynamic logic), in which the logical (declarative) state descriptions are assigned in any progress of a program from an initial state to the state when the program terminates.

3.2 Formal verification for program correctness and explainability

When examining the explanatory nature of program software in a B-level environment, the first thing to realize is that if a logical environment is added, optimistic predictions can be made from the standpoint of various theories of "formal verification methods" of program correctness.

One logical way is to use the Curry-Howard isomorphism theorem, in which a program in a typed functional programming language or its extension to a type theory has a logical correspondence with a formal logical proof (cord). For ordinary procedural program languages, we can mention an alternative logical way, namely, Hoare logic and its extension, dynamic logic.

- Facts about logical program verification theory:
  The Curry-Howard isomorphism can identify a logical proof and a typed functional program, while Hoare logic (and dy-

namic logic) can describe a prodedural program as the logical states transision process.
- In addition, the same is true to a certain extent for:
  The executable algebraic specification languages, and the logic programming languages such as prolog.

The interpretation of Hoare logic proofs in terms of the state-transitive relation between Hoare logic proofs and programs can be considered.[3] A more direct correspondence between programs and proofs is found in the case of typed functional languages.

First, let us mention the origin of the abstract programming language untyped lambda calculus. J. McCarthy implemented a symbolic computation processing language called lisp based on untyped lambda calculus in the 1960s, which was the world's first AI language (AI here is first-generation symbolic AI). He is also known for being the first to use the term Artificial Intelligence.

Typed lambda calculus was given by Church-Kleene. The correspondence between typed lambda calculus and logic is as follows.

- Type: Logical formula
- Program (lambda term): Proof
- Program execution: Proof normalization (cut rule elimination)

This correspondence is called the Curry-Howard isomorphism theorem. Although the logic used for typed lambda calculus was limited, Martin-Löf and others enriched it, which is called type theory. Implementations of proof assistant and verification tools based on type theory include Coq, Isabelle, Agda, Hol, Lean and others.

We specifically refer to Coq (https://rocq-prover.org/). In this subsection, we mainly consider the formal verification view of programs. We will focus on proof constructions with these proof assistant tools.

Assume that in a B-level environment where semi-automatic judgment programs have been used for loan applications by financial institutions, where judgments for loan applications are made by programs based on type theory. When a logic-proof analysis of loan review programs becomes possible, it should be possible, at least theoretically, to provide a proof for loan review decisions. This would seem to indicate that explainability is at hand in the B-level environment, compared to the difficulties with explanability currently faced by judges based on deep learning and big data, in the C-level environment.

What is meant by explainability is the ability to explain a decision on an on-demand basis when an explanation is requested and needs to be provided. For example, suppose a loan application is denied. In an A-level environment, a financial institution's examiner can testify as to how and at what point in the process, he or she made the determination of ineligibility in accordance with his or her manual provided by the financial company. On the other hand, in a B-level environment, if a loan application is automatically reviewed by type theoretic software and a loan application is determined ineligible, the decision software can provide a logical proof of ineligibility. The financial institution would be able to logically prove in court that the loan applicant is ineligible to be granted the loan. However, what is the loan applicant's appeal of the software's logical denial of the loan application? It may not be an appeal against the argumentative rational-logical decision-making process, but rather a disagreement with the premises of the proof (i.e., the axioms and definitions assumed in the proof). This point is discussed further elsewhere.

If one were to ask whether decisions were ever made in a B-level environment solely by type-theoretic programs in the screening of loan applications, the answer would be in the negative. However, since 1990, a major Japanese money and consumer loan institution launched a service called "MUJIN-KUN," in which a loan/credit card application and an automatic response to the screening result could be obtained almost immediately at an ATM with no human presence on the financial company side, giving general consumers the image of software automatic screening in Japan. In reality, it is assumed that the applicant's information is located at the financial institution's credit card screening department via the internet, and only semi-automatic software screening is conducted. Therefore, it is not possible to provide proof of privacy and security of the applicant's information to the "MUJIN-KUN" screening pro-forma. However, for now, please consider this as a thought experiment to see how explainability would be theoretically possible if fully automated software screening were conducted.

Type-theoretic languages have been implemented and used in practice to verify whether a program is guaranteed to compute to specification. This is to show whether or not the type of a specification can be obtained through a series of type inferences that compose the program. Verifying the specification type through type inference means verifying the correctness of the program, where the correctness of the program usually includes the condition "if the execution of the program terminates," since program termination is guaranteed in a wide range of programming languages based on type inference, this is a verification of the correctness of the program, including the halting property of the program. Here, the specification is given as a logical expression, which corresponds to a type. Verification that a given program is correct according to the specifications corresponds to checking that this program is a proof of the logical formula expressed by the specification.

Now we focus more on proofs to ask to what extent a proof

is explanatory further and and what would be the issues on explainabiliy of proofs in the context of proof assistant systems.

3.3 A gap between formal verification and explanatory proofs

Even if software program verification corresponds in principle to logical proof verification, proof by program verification in reality is not always sufficiently explanatory.

Even if it can be verified through type inferences in a typed language that a program is typed as specified, it is not known whether the logic proof code formed in the process can be viewed as an explanatory proof. A gap is expected to remain in this regard. The specification is given in the form of a proposition (logical formula), and program verification involves verifying whether or not the program has this type. In other words, it is to verify that the program corresponds to the proof of the logical formula that represents this specification. As far as program verification is concerned, the interest is in using programs that are guaranteed to be correct, and not in interpreting programs as proofs. Type-theoretic verification systems are often called proof assistant systems. They have the role of assisting the user to construct proofs. However, if the main aim is the program verification, attention will not be paid to the proof interpretation itself.

Thematizing the proof itself, which is constructed along with the logical form, and asking whether it is also explanatory as a proof, was often a secondary issue in the framework of 20th century formal verification.

Let us look back at the fact that type theory tools are called proof assistant systems and were originally designed to assist proof construction. Proof assistant systems based on type theory have as their primary purpose to construct proofs using formal logical reasoning steps interactively with a user. They often also have partially automatic proof functions. The formalization of mathematical proofs to verify the correctness of proofs is a well-known major advantage of these type-theoretic proof assistant systems. Here are just a few examples, many of which constitute rigorous proofs consisting of reasoning steps formalized in a logical formal language.

The four-color problem conjecture, which asserts that any map can be painted in four colors, was claimed to have been solved affirmatively by the use of computers in the 1970s, when computers became available. Later, in the 1990s, a more compact 1482-case proof was given by computer due to improvements in some errors, and in the 1970s, confidence in the correctness of the positive proof of the four-color problem was relative to the large number of computer checks. In 2004, Coq formalized the entire proof and verified its correctness in formal logic. What this means is that a logically error-free formal proof has been constructed with Coq. The proof code can in principle be presented as a formal logical proof. However, if one were to literally print or diplay the Coq code as a single proof with Curry-Howard correspondence, the formalization on Coq would also involve the translation of many non-essential code parts into a proof. For the Coq type inference system, the proof is error-free, but the whole proof of the four-color theorem will not be understandable and graspable, and the proof will not be perceived as explanatory.

3.4 A gap between formal logical proofs and explanatory proofs

Based on the formalization practices of Coq and other major type-theoretic proof assistant systems and mathematical theories, we can say the following. Although it can produce a proof code, the proof code itself cannot be regarded as an overall explanatory proof. This is similar to the situation when proof assistant systems are used for software verification.

The following is a list of the most common problems:

1. Generally speaking, even details that are not the main point of the proof are translated into the proof.
2. As is true of formalized proofs in general, they are often long, complex, and hard to survey.
3. Proof codes and proof expressions are difficult to read for ordinary people who are familiar with reading proofs in a natural language or in an ordinary logical and mathematical language.

Regarding point 3, it is possible to resolve this by coordinating the reasoning system of the C-level environment, as typified by Large Language Models (LLMs), with the proof assistant system of the B-level environment. It is important to see that the gap described by 3 is the gap between the proof assistant's proofs and the readable proofs in natural language or in ordinary logical or mathematical language, which has been situated at the A-level . Bridging the B-level and the C-level via LLMs also means bridging with the A-level.

Regarding point 1, we can expect theoretical improvements in the proof assistant system itself. On the other hand, it is possible to construct a rigorous formal proof and at the same time remove non-essential parts from type inferences. For example, in Coq, an attempt was made to handle the equational transformation part with a term rewriting proof system. Such an example suggests that the type-reasoning proof paradigm can be slimmed down by combining it with other proof calculation paradigms.

Regarding point 2, as an extension of the solutions to 1 and 3, we can believe that interactive coordination between proof assistant systems and LLMs will offer possible improvements. In the case of proof construction of known theorems, it will depend on the ability of proof assistant systems and LLMs to construct libraries of lemmas. In the case of known theorems, both proof assistant systems and LLMs often have large can-

didate lemma data. However, the issue that remains is the explanatory nature of the proofs.

Dowek [4] holds that an explanatory proof should be an explanation that reveals "reasons." Dowek appears to suggest that a proof of a proposition more general than the target expected proposition may explain why the target proposition holds. It also appears that a proof that gives a generalized property, and is a special case from it, has explanatory properties. For example, an explanatory proof of why $3 \times 5 = 5 \times 3$ or $473 \times 23 = 23 \times 473$ such as concrete numerical examples of the exchange law of multiplication are valid by giving a proof of the general proposition "For all a and b, $a \times b = b \times a$" for which these two equalities are special cases, The above equation can be thought of as a proof of the general proposition "For all a and b, $a \times b = b \times a$."

In such an explanation of $3 \times 5 = 5 \times 3$, the proof of the general exchange rule of multiplication may play the role of "reason" and has a certain commonality with Dowek's idea. On the other hand, for Wittgenstein, although he acknowledges the proof structure of mathematical induction ("the uniqueness rule" in his formulation), he rejects to state $a \times b = b \times a$ as the conclusion of a mathematical induction. For all a and b, $a \times b = b \times a$ is unnecessary to say in order to conclude $3 \times 5 = 5 \times 3$ or $473 \times 23 = 23 \times 473$, according to Wittgenstein. Thus, there is a disagreement. This article basically follows Dowek's position; the Wittgensteinian explanation will be discussed at another time.

Let us return to the formalization and verification by Coq of the proof of the four-color problem. Regarding the gap between proofs generated by proof assistant systems and explanatory proofs, we noted above that they could be closed. However, even in the formalization by Coq of the four-color problem proofs, there is still a 1482-case structure. From the standpoint of explaining the reason by generalization, it would be required to prove a single proposition of a more general nature and then to construct a structure that intuitively shows that these 1482 cases are special cases. This is unlikely to be so, as Coq's formal verification of the four-color problem shows, but it may be an example of the difficulty of presenting an explanatory proof. It is important to note that Coq's formalization has successfully verified provability of the four-color theorem. However, the proof obtained do not provide us with a proof that we can understand as explanatory.

Suppose we have a consumer loan approval system that relies heavily on software. Suppose the software has proof expressions using logical methods. Suppose now that the summary of the proof representation is human readable in natural language using the methodology sketched above. Suppose that proof is given that Ms. A's loan application as submitted does not meet the criterion; the general provability criterion that Dowek suggests might be interpreted as involving some kind of fairness: instead of proof that only applies to Ms. A in a discriminatory way, it might be interpreted to mean that all people in a certain condition are entitled to a loan without meeting the criteria. After it is proven that they do not meet the criteria and cannot receive a loan, the proof would then take the form of an explanation that Ms. A also cannot receive a loan because Ms. A is an instance of the people.

3.5 Large Language Models

We have presented the idea of using a combination of proof assistant systems and Large Language Models in a cooperative manner for producing explanatory proofs. I would like to take this opportunity to introduce some of our group's data-centric AI research on formal reasoning and explanations of formal inference. Here, we are specifically evaluating reasoning ability of LLMs and building datasets.

The typical version of Peirce's abdactive reasoning is to infer a hidden explanatory hypothesis in the syllogistic inference format (where the conclusion and the major hypothesis are explicit). When LLMs were used to evaluate the reasoning ability of the Abduction tasks, some of the LLMs performed well enough.[5] Although this is only one of the initial studies, we note that the ability of LLMs to handle reason-conclusion relationships appears to be similar to that of humans. It is also known that LLMs' reasoning is prone to errors due to human psychological biases, such as those represented by cognitive biases, even when limited to syllogistic reasoning and plain propositional logical reasoning tasks. On the other hand, our group set up a prompt to have LLMs respond to a logical reasoning task while explaining the path to the conclusion of the task in the form of a predicate logical terms. The results showed that the model tends to be less prone to cognitive bias and the number of correct responses generally increases when a logical explanation is provided.[6]

3.6 An example of logical summarized proof of computational proof from the B-level network environments and logical formal verification

The network environment is one of the hallmarks of the B-level environment, and it often appears that probabilistic and statistical elements that go beyond symbolic processing-based formal verification are becoming more prominent in the formal verification or formal method paradigm for the network environment. Needless to say, the focus is on what goes beyond symbolic computation in the C-level machine learning and big data environments.

Security verification of network communications is often performed using logical verification methods as an extension of software security verification. This includes security (authentication, secrecy, etc.) of cryptographic protocols that are appropriate for the purpose. There have been two major methods for security protocol verification. One is a formal method and the other is a cryptography-based computational

method. For example, in our group's research on security protocol verification, a series of message exchanges along a protocol can be expressed in terms of logical proofs. On the other hand, in the cryptology-based method, security proofs are based on, for example, through discussions of the computational model of a probabilistic Turing Machine. If the formal verification of security protocol at the level of symbolic logic proofs is successful, the question of whether it can be said to be secure in the cryptographic-computational model that is closer to the real world (than the logical symbolic model) is a problem related to the reliability of formal verification (logic proof) method, and this question is called the computational soundness problem of formal verification. Our group's results [7] include an affirmative proof of computational soundness as well as verification methods for the case in which a notion of compatibility is primitive and for the case in which indistinguishability is the primitive. Proofs of security properties by logical proof method are much simpler and explanatory than those by the cryptographic computational method. The logical proofs by the formal method are symbolic summarized proofs of the computational proofs for the computational real world. However, it is guaranteed that the computational details can be ignored once the computational soundness is established. In fact, computational soundness is established. Hence, we can work on easily understandable explanatory logical proofs. As already discussed, the notion of summarized proof is sometimes needed for proofs to be explanatory. The example of summarized proofs given here may be instructive when further investigation on summarized proofs is needed.

Security-protocols for e-voting via network have been studied for a long time to support secure e-elections. On the other hand, we have heard that there has been a proposal in the past for an empiricist-practical position that differs from the traditional rational-logical position. The proposal is that if the results of electronic voting are counted on the servers of each political party independently and the results of the winners and losers do not differ, then minor discrepancies in the results can be ignored and the vote count can be considered to have been final. The fact that exit polls using AI sometimes fail to predict the outcome of elections (who is the winner) seems to indicate the necessity of taking into account the logical position and method for secure e-voting.

4. Basic research toward a fair information presentation environment that supports people's independent decision-making

Departing from the motivation of this article so far, which has been to question the reasons and explanations for decisions by software and AI, we now consider the situation that we are in a position to make a decision. We would like to ask what kind of information presentation does not contain bias or manipulation and facilitates proactive decision making.

Fake news, fake information, and behavioral targeting ads through the internet and AI environment—all these are overflowing. They are major bias factors in our individual and group decision making, and are also involved in fraud and other crimes.

As already indicated in the Introduction, the purpose of this article is not to technically discuss solutions to the problems of the current C-level environmental society itself, it is to find common problems with those problems from the human procedural level (A-level) societies with software and network procedural level (B-level), and to discuss some aspects of the underlying factors of the problems of today's digital and AI-environmental society.

As I have already indicated, my position in this article is that some of the essential sources of the ethical issues in the C-level environment are already present in the A-level and B-level environments, and that it is meaningful to study C-level issues from a new perspective by reexamining the issues already present in A-level and B-level environments. Here, I refer to a basic research example of multi-attribute decision making, which is a form of decision making in which multiple options and their attributes are presented on a single matrix table (a multi-attribute table) on paper or on a screen, and decision makers make choices based on the information presented in the table. A typical multi-attribute choice table would be a product catalog that contains multiple products and multiple attribute information for each product (such as a catalog of digital cameras). A paper-based product catalog is a typical example of primitive information presentation in an A-level environment. Its e-commerce version is the primitive counterpart of B-level information presentation. We refer to our research on decision-making information presentation design and online information presentation design.

Here, we would like to focus on "a single viewability" concept. For example, Hino evaluates the system of "a single view" public election poster boards (see Airo Hino, in August 13, 2024, Chuo-Koron for the exact argument). It has been followed since the (A-level environment in our sense) society, as supporting the three principles of democratic election systems: Openness, Fairness, and Legitimacy. We live in an era in which voters can obtain much of their information about a particular candidate or party from their own political perspective via the internet (B-level environment). Under these circumstances, some argue that there is no need for public election poster boards. In addition, [along with current AI technology (C-level environment)], targeted advertising and biased information on social networking sites that suits individual political preferences can become exclusively accessible, a situation known as filter bubbles. On the other hand, the election poster boards (or official election pamphlets) have significance in a democratic election system, as they provide an opportunity to compare and review the policies of other

parties and basic demographic information of all candidates in a given constituency, according to Hino.[8]

In our research on the design of multi-attribute, multi-choice product catalog presentation, we use an eye tracker method. We aim to clarify how product catalog design can help decision makers make decisions more easily.[9] We believe that we can gain insight into the manipulative and inductive designs. This research is rooted in the basic idea that fair presentation is information presentation design that allows decision makers to make decisions proactively. In reality, we believe that the criteria for fair presentation design will change for each domain of the multi-attribute table. For example, in the domain of product catalogs as one-view information in consumer behavior, our experimental results show that highlighting good attribute values to make them stand out is one of the fair presentation designs. We found that the vertical alignment of choices and the horizontal alignment of multiple attributes tends to make decision making easier than a weakly aligned product catalog. The role of the vertical and horizontal axes is seen in both cases for paper-based product catalogs in the A-level environment. On the other hand, in the B-level environment of e-commerce, product options are overwhelmingly arranged on the vertical axis and multiple attributes are arranged on the horizontal axis, which is consistent with an information presentation design suitable for decision making. This is consistent with a decision-oriented information presentation design. For example, a decision maker may focus on multiple attributes of interest, narrow down a small number of items that have a trade-off relationship among many items, and present a one-view-catalog of his/her order-made products to make a trade-off-aware decision. If this final-stage catalog is highlighted as shown above, it will be easier to make a decision at the highest level. By analyzing data at the eye-movement level, we can examine the design of product catalogs that are decision-friendly for decision-makers through trends in the relationship between the design of information presentation and decision-making strategies. While early decision making has often been considered easy, a two-step strategy of narrowing down the choices and then considering trade-offs among a small number of options is often a better decision-making strategy. Although the product catalog is only one example, this article shows that it is possible to analyze the effects of fair information presentation and unfair (e.g., manipulative, inducive) tabular presentation on decision making only in the primitive setting of one-view information presentation. I believe that this study provides basic information for examining the fairness of information presentation in a C-level environment.

Notes

(1) A dynamic and global annual celebration of World Logic Day aims at fostering international at the dawn of this new decade—indeed, now more than ever—the discipline of logic is utterly vital to our societies and economies. Computer science and digital technology, which provide the structure for today's ways of life, are rooted in logical and algorithmic reasoning. Computer science and digital technology, which provide the structure for today's ways of life, are rooted in logical and algorithmic reasoning. Audrey Azoulay, Director-General of UNESCO.

(2) UNESCO's first-ever global standard on AI ethics—the 'Recommendation on the Ethics of Artificial Intelligence,' adopted in 2021, is applicable to all 194 member states of UNESCO. Recommendation on the Ethics of Artificial Intelligence UNESCO 2022.

(3) Cf. Tennent, R. D. (2002). *Specifying software*. Cambridge University Press. Let S be a part of a program; the state of S before its execution is called the pre-condition of S, and the state of S after its execution is called the post-condition of S. The pre-condition and post-condition of S are generally expressed by logical expressions in predicate logic. That "if P holds before the execution of S, then Q holds after the execution of S." In symbol, P {S} Q, and P and Q are the pre-condition and post-condition of S, respectively. We will not go into the technical details of Hoare logic here, but note that the progress of the execution process of a program is represented by a series of transitions of declaratively described logical states, from the initial state to the final state.

(4) Dowek, G. (2023). From ethics to logic. *Annals of the Japan Association for Philosophy of Science*, Vol. 32, pp. 1-16. The author's standpoint in this article is not always shared with Dowek, but the author's argument is very much inspired and influenced from his.

(5) Abe, H., Ozeki, K., Ando, R., Morishita, T., Mineshima, K., and Okada, M. (2024). Abductive reasoning with syllogistic forms in large language models. *Proceedings of Human and Artificial Rationalities*, pp. 3-17.

(6) Ozeki, K., Ando, R., Morishita, T., Abe, H., Mineshima, K., and Okada, M. (2024). Exploring reasoning biases in large language models through syllogism: Insights from the NeuBAROCO dataset. *Findings of ACL*, pp. 16063-16077.

(7) Bana, G. and Okada, M. (2016). Semantics for "enough-certainty" and fitting's embedding of classical logic in S4." *Computer Science Logic*, Vol. 34, pp. 1-34. Bana, G., Chadha, R., Eeralla, A. K., Okada, M. (2019). Verification methods for the computationally complete symbolic attacker based on indistinguishability. *ACM Transactions on Computational Logic*, Vol. 21, No. 2, pp. 1-44.

(8) Hino, A. (2024). *Tochiji-sen 'hakku' de towa reru Nihon no senkyo* [Japan's elections at stake in the Tokyo gubernatorial election in district 8]. *Chuo-Koron*, September issue (in Japanese).

(9) Morii, M., Ideno, T., Tamari, Y., Takemura, K., and Okada, M. (2024). An eye-tracking study on the effects of using high-

lighted multi-attribute tables: A preliminary report. *Proceedings of Diagrammatic Representation and Inference*, pp. 467-471. Morii, M., Takemura, K., and Okada, M. (2020). On the effects of changing multi-attribute table design on decision making: An eye-tracking study. *LNAI*, Vol. 12169, pp. 365-381." Morii, M., Ideno, T., Takemura, K., and Okada, M. (2017). Qualitatively coherent representation makes decision: Making easier with binary-colored multi-attribute tables: An eye-tracking study. *Frontiers in Psychology*, Vol. 8, 1388.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 21H00467, 23K20416, and 21K18339.

Published: June 30, 2025

https://doi.org/10.11425/sst.14.5