Foundations of Reasoning with Uncertainty via Real-valued Logics

Ronald Fagin[a,1], **Ryan Riegel**[a], **and Alexander Gray**[a]

[a]IBM Research

This manuscript was compiled on April 7, 2024

Interest in logics with some notion of real-valued truths has existed since at least Boole, and has been increasing in AI due to the emergence of neuro-symbolic approaches, though often their logical inference capabilities are characterized only qualitatively. We provide foundations for establishing the correctness and power of such systems. We introduce a rich, novel class of multidimensional sentences, with a sound and complete axiomatization that can be parametrized to cover many real-valued logics, including all the common fuzzy logics, and extend these to weighted versions, and to the case where the truth values are probabilities. Our multidimensional sentences form a very rich class. Each of our multidimensional sentences describes a set of possible truth values for a collection of formulas of the real-valued logic, including which combinations of truth values are possible. Our completeness result is strong, in the sense that it allows us to derive exactly what information can be inferred about the combinations of truth values of a collection of formulas given information about the combinations of truth values of a finite number of other collections of formulas.

We give a decision procedure based on linear programming for deciding, for certain real-valued logics and under certain natural assumptions, whether a set of our sentences logically implies another of our sentences. The generality of this work, compared to many previous works on special cases, may provide insights for both existing and new real-valued logics whose inference properties have never been characterized. This work may also provide insights into the reasoning capabilities of deep learning models.

Keywords: real-valued logic | finite-strongly complete axiomatization

Formalization of the idea of *real-valued logics* (a term which is perhaps not standard but we will use to refer to various proposals that extend classical logics to ones where truths can take arbitrary values in the range $[0, 1]$) is old and fundamental, going back to the origins of formal logic. It is not well known that Boole himself invented a probabilistic logic in the 19th century (1), where formulas were assigned truth values corresponding to probabilities. It was used in AI to model the semantics of vague concepts for commonsense reasoning by expert systems (2). Real-valued logics have appeared in linguistics to model certain natural language phenomena (3), in hardware design to deal with multiple stable voltage levels (4), and in databases to deal with queries that are composed of multiple graded notions, such as the redness of an object, that can range from 0 (“not at all red”) to 1 (“completely red”) (5). Despite all this, while definitions of logical correctness and power (generally, soundness and completeness) are well established and corresponding procedures for *theorem proving* having those properties are abundant for classical logics, the equivalents for real-valued logics are comparatively limited. Though some formal properties have been established for certain special cases of real-valued logics, the analysis is typically delicate in that it cannot easily be extended if the logic is extended or changed, or may only show weaker properties than possible. We discuss previous works in Section 9.

Recent years have seen growing interest in AI in approaches for augmenting the capabilities of learning-based methods with those of reasoning, often broadly referred to as *neuro-symbolic* (though they may not be strictly neural). One of the key goals that neuro-symbolic approaches have at their root is logical inference, or reasoning. However, the representation of classical 0–1 logic (where truth values of sentences are either 0, representing “False”, or 1, representing “True”) is generally insufficient for this goal because representing uncertain knowledge and conclusions is essential to AI. In order to merge with the ideas of neural learning, the truth values dealt with must be *real-valued* (we shall take these to be real numbers in the interval $[0, 1]$, where intuitively, 0 means “completely false”, and 1 means “completely true”), whether the uncertainty semantics are those of probabilities, subjective beliefs, neural network activations, or fuzzy set memberships. For this reason, many major approaches have turned to real-valued logics. Logic tensor networks (6, 7) define a logical language on real-valued vectors corresponding to groundings of terms computed by a neural network, which can use any of the common real-valued logics (e.g., Łukasiewicz, product, or Gödel logic) for its connectives (e.g., &, $\veebar$, $\neg$, and $\rightarrow$). Probabilistic soft logics (8) draw a correspondence of their approach based on Markov random fields (MRFs) with satisfiability of statements in a real-valued logic (Łukasiewicz). Tensorlog (9), also based on MRFs but implemented in neural network frameworks, draws a correspondence of its approach to the

Significance Statement

This work introduces a rich, novel class of multidimensional sentences that yield a new sound and complete axiomatization for a larger class of real-valued logics than previously considered, including all of the most common fuzzy logics, weighted versions, and probabilistic logics, many of which have garnered renewed interest as a result of the developing field of neuro-symbolic AI. Here, “complete axiomatization” holds in a strong sense: whenever a finite set Γ of our sentences logically implies one of our sentences γ, that is, whenever every model of Γ is a model of γ, then there is a proof of γ from Γ using our axiomatization. A decision procedure for two of the popular such logics, under certain natural assumptions, is presented. This work may also provide insights into the formal reasoning capabilities of deep learning models.

[1]To whom correspondence should be addressed. E-mail: fagin@us.ibm.com

use of connectives in a real-valued logic (product). Logical Neural Networks (LNN) (10, 11) represent a methodology which draws a correspondence between activation functions of neural networks and connectives in real-valued logics. To complete a full correspondence between neural networks and statements in real-valued logic, LNN defines a class of real-valued logics allowing weighted inputs, which represent the relative influence of subformulas. This follows the earlier observation of this connection between neural networks based on rectified linear units (ReLU) and weighted real-valued logics in (12). Notably, work on large language models based on such networks has shown anecdotal examples that appear to indicate the capability of sometimes-successful reasoning, though the extent and underlying mechanisms still remain open mysteries. While widely regarded as fundamental to the goal of AI, the reasoning capabilities of the aforementioned systems are typically made qualitatively versus quantitatively and mathematically. While learning theory (roughly, what it means to perform learning) is well articulated for a large class of models and, for 0–1 logic, what it means to perform reasoning is well studied, reasoning is surprisingly not well formalized for a large class of real-valued logics. As reasoning becomes an increasing goal of learning-based work, it becomes important to have a solid mathematical footing for it.

Soundness and completeness. In this paper, there are two levels of logic. In the "inner" layer, we have formulas of the real-valued logic with its logical connectives. In particular, in this inner layer, we shall use & for "and" and $\veebar$ for "or", as is done in (13). In the "outer" layer, we have a novel class of multi-dimensional sentences about the inner real-valued logic, such as saying which truth values a given real-valued formula may attain, or even more, what combinations of values several real-valued formulas may attain. For these sentences in the outer layer, which take on only the classical values 0 and 1 for False and True, respectively, we in particular make use of the traditional logical symbols $\wedge$ for "and" and $\vee$ for "or". We remark that, somewhat confusingly, the symbols $\wedge$ and $\vee$ are often used in real-valued logics for weaker versions of "and" and "or" than that given by & and $\veebar$, which we do not have need to discuss in this paper.

Let us say that an axiomatization of a logic is *finite-strongly complete* if whenever Γ is a finite set of sentences in the (outer) logic and γ is a single sentence in the (outer) logic that is a logical consequence of Γ (that is, every model of Γ is a model of γ), then there is a proof of γ from Γ using the axiomatization. An axiomatization is *weakly complete* if this holds for $\Gamma = \emptyset$. That is, an axiomatization is weakly complete if whenever γ is a valid sentence (true in every model), then there is a proof of γ using the axiomatization. The reader might think we can obtain a finite-strongly complete axiomatization from a weakly complete axiomatization by believing that if φ_1 logically implies φ_2, then the formula $\varphi_1 \rightarrow \varphi_2$ is valid. This is true for Gödel logic (as noted in (14); see also (13)), but it is false for Łukasiewicz logic. A counterexample in Łukasiewicz logic is obtained (as the reader can easily verify) by taking φ_1 to be the formula A and φ_2 to be the formula $A \,\&\, A$.

Early axiomatizations of real-valued logics in the literature were typically weakly complete, but now have often been improved to finite-strongly complete. As Di Nola and Lettieri point out in their paper on a normal form for Łukasiewicz logic (12), Rose and Rosser (15) gave a syntactic proof of weak completeness for an axiomatization of Łukasiewicz logic, and later Chang gave an algebraic proof (16, 17). Hájek and Svedja (14) later gave a finite-strongly complete axiomatization. There is also a finite-strongly complete axiomatization for Gödel logic (18). In Section 3, we shall show why it is necessary to assume that Γ is finite in the definition of finite-strongly completeness. From now on (except in the Section 9 on related work) we use "complete" to mean "finite-strongly complete".

All previous axiomatizations we have discussed so far deal only with formulas, and not with the truth values assigned to formulas. Thus, they may infer when a formula γ follows from a finite set Γ of formulas (that is, whether γ necessarily has truth value 1 when every formula in Γ has truth value 1), but not whether a certain arbitrary truth value or set of possible truth values for γ can be inferred from information about the possible truth values of members of Γ. A limited form of such inference can be done for Łukasiewicz real-valued logic by combining it with rational Pavelka logic (see Section 9 for a discussion on this).

This paper. We introduce a rich, novel class of multidimensional sentences ("MD-sentences") with a sound and complete axiomatization.

1. These sentences can say what the set S of possible values is for a formula σ. This set S can be a singleton $\{s\}$ (meaning that the truth value of σ is s), or S can be an interval, or a union of intervals, or in fact an arbitrary subset of $[0,1]$, e.g., the set of rational numbers in $[0,1]$.

2. Our sentences can give not only the possible truth values of formulas, but the interactions between these values. For example, if σ_1 and σ_2 are formulas, our sentences can not only say what the possible truth values are for each of σ_1 and σ_2, but also how they interact: thus, if s_1 is the truth value of σ_1 and s_2 is the truth value of σ_2, then there is a sentence in our logic that says (s_1, s_2) must lie in the set S of ordered pairs, where S is an arbitrary subset of $[0,1]^2$.

3. Unlike the other axiomatizations mentioned earlier, our axiomatization can be extended to include the use of weights for subformulas (where, for example, in the formula $\sigma_1 \veebar \sigma_2$, the subformula σ_1 is considered twice as important as the subformula σ_2).

A surprising feature of our axiomatization is that it is parameterized, so that this one axiomatization is sound and complete for a large class of real-valued logics including all of the most common fuzzy logics and even logics that do not obey the standard restrictions on fuzzy logics (such as conjunction being commutative). Previous axiomatizations in the literature required a separate set of axioms for each real-valued logic (for example, one of the axioms for Łukasiewicz logic is $\sigma \leftrightarrow \neg\neg\sigma$, and one of the axioms for Gödel logic is $\sigma \leftrightarrow (\sigma \,\&\, \sigma)$). Such axiomatizations correspond to fixed truth evaluation functions associated with each connective. By contrast, for our axiomatization, evaluation functions may be arbitrary, where k-ary connectives map $[0,1]^k$ into $[0,1]$.

In fairness and giving credit to the completeness results in the literature for various real-valued logics, it should be noted that since our MD-sentences are much more expressive than those logics, the soundness and completeness for our parametric axiom system for MD-sentences does not supersede or entail soundness and completeness results for less expressive systems. Showing that a proof system featuring only modus ponens and a number of axiomatic formula schemes is (sound and) complete for a specific logic is, in general, a much harder task than we faced, where we could make use of the vast generality of one of our inference rules (Rule 7 below).

Throughout this paper, we take the domain of each function in the real-valued logic to be $[0,1]$ or $[0,1]^2$ and the range to be $[0,1]$. This is a common assumption for many real-valued logics, but all of our results go through with obvious modifications if the domains are D^k for possibly multiple choices of arity k and range D, for arbitrary subsets D of the reals. We note that real-valued logic can be viewed as a special case of multi-valued logic (19), although in multi-valued logic there is typically a finite set of possible truth values, not necessarily linearly ordered.

We also provide a decision procedure for deciding whether a set of our sentences logically implies another of our sentences for certain common real-valued logics under certain natural assumptions. We implement the decision procedure, dubbed **SoCRAtic** (for **So**und and **C**omplete **R**eal-valued **A**xioma**tic** solver), which we describe in detail in Section 6. While our sentences allow a wide variety of real-valued logics, as does our sound and complete axiomatization, this decision procedure depends heavily on the choice of logical connectives and in particular is tailored towards Łukasiewicz and Gödel logic, though it can be adapted to support product logic as well.

Overview. In Section 1, we give our basic notions, including what a model is and what a sentence is. In Section 2, we give our (only) axiom and our inference rules. In Section 3, we give our soundness and completeness theorem. In Section 4, we give a theorem that says that each finite Boolean combination of our sentences is equivalent to a single one of our sentences which helps to show the robustness of our class of sentences. In Section 5, we discuss possible reductions of the dimensionality of our sentences. In Section 6, we give the decision procedure. In Section 7, we show how to extend our methodology to incorporate weights. In Section 8, we discuss how to deal with treating the truth values as probabilities. In Section 9, we discuss related work. In Section 10, we give our conclusions and review their implications for AI approaches.

1. Models, formulas, and sentences

We assume a finite set of atomic propositions. These can be thought of as the input layer of a neural net, i.e., nodes with no inputs from other neurons. A model M is an assignment g^M of truth values to the atomic propositions. Thus, M assigns a value $g^M(A) \in [0,1]$ to each atomic proposition A.

We now define the set F of logical formulas. For simplicity, we assume for now that there are just four logiclal connectives: three binary connectives, namely conjunction (denoted by &), disjunction (denoted by $\veebar$, and implication (denoted by $\rightarrow$), and one unary connective, namely negation (denoted by $\neg$). However, our definiitions and results extend easily to arbitrary sets of logical connectives of arbitrary arity.

The set F of logical formulas is defined inductively. Every atomic proposition is a logical formula. If σ_1 and σ_2 are logical formulas, then so are (a) $\sigma_1 \,\&\, \sigma_2$, (b) $\sigma_1 \veebar \sigma_2$, (c) $\sigma_1 \rightarrow \sigma_2$, and (d) $\neg\sigma_1$.

Two especially useful real-values logics for logical neural networks are Łukasiewicz logic and Gödel logic. Let σ_1 and σ_2 be formulas with respective truth values s_1 and s_2. For Łukasiewicz logic, the truth value of $\sigma_1 \,\&\, \sigma_2$ is $\max\{0, s_1 + s_2 - 1\}$, the truth value of $\sigma_1 \veebar \sigma_2$ is $\min\{1, s_1 + s_2\}$, the truth value of $\sigma_1 \rightarrow \sigma_2$ is $\min\{1, 1 - s_1 + s_2\}$, and the truth value of $\neg\sigma_1$ is $1 - s_1$. In Gödel logic, the truth value of $\sigma_1 \,\&\, \sigma_2$ is $\min\{s_1, s_2\}$, the truth value of $\sigma_1 \veebar \sigma_2$ is $\max\{s_1, s_2\}$, the truth value of $\sigma_1 \rightarrow \sigma_2$ is 1 if $s_1 \leq s_2$ and s_2 otherwise, and the truth value of $\neg\sigma_1$ is 1 if $s_1 = 0$ and 0 otherwise.

If α is a binary connective, then by $f_\alpha(s_1, s_2)$ we mean the value of $\sigma_1 \,\alpha\, \sigma_2$ if the value of σ_1 is s_1 and the value of σ_2 is s_2. For example, in Łukasiewicz logic, $f_\&(s_1, s_2)$ is $\max\{0, s_1 + s_2 - 1\}$. For the unary connective $\neg$, by $f_\neg(s_1)$ we mean the value of $\neg\sigma_1$ if the value of σ_1 is s_1. For example, in Łukasiewicz logic, $f_\neg(s_1)$ is $1 - s_1$.

We now define by induction on the structure of formulas what the truth value of a formula in F is in a model M, for a given real-valued logic. By definition of a model, we know the truth value in M of an atomic proposition. If α is a binary connective then the truth value in M of $\sigma_1 \,\alpha\, \sigma_2$ is $f_\alpha(s_1, s_2)$ if the truth value in M of σ_1 is s_1 and the truth value in M of σ_2 is s_2. The truth value in M of $\neg\sigma_1$ is $f_\neg(s_1)$ if the truth value in M of σ_1 is s_1.

When considering only formulas with truth value 1, as is common when giving an axiomatization of a real-valued logic, the convention is to consider a sentence to be simply a member of F. What if we want to take into account values other than 1? It is tempting to think we can simply annotate formulas with truth values or sets of truth values, for instance with sentences of the form $(\sigma; S)$ where $\sigma \in F$ and $S \subseteq [0,1]$, which indicates the truth value of φ is in S. In fact, we note that formulas equivalent to $(\sigma; S)$ have been considered in the literature (20, 21) in the special case where S is an interval. Our sentences go a step further and annotate *groups* of formulas with sets of tuples of truth values.

We take a sentence γ to be an expression of the form $(\sigma_1, \ldots, \sigma_k; S)$, where $\sigma_1, \ldots, \sigma_k \in F$ are the *components* of γ and where $S \subseteq [0,1]^k$ is the *information set* of γ. The intuition is that $(\sigma_1, \ldots, \sigma_k; S)$ says that if the value of each σ_i is s_i, for $1 \leq i \leq k$, then $(s_1, \ldots, s_k) \in S$. Also S may contain other tuples of truth values (some possibly inconsistent, such as having the value of $A \,\&\, B$ being strictly higher than the truth value of A). Inference then proceeds to form new sentences with restricted sets S in attempt to identify the s_i for $1 \leq i \leq k$ or, alternatively, prove that none can exist.

Note that we are not restricting the information sets S to be simple, such as being the union of a finite number of intervals (and in the case of Łukasiewicz or Gödel logic, for the intervals to have rational endpoints). Such a restriction is common in the literature for sentences $(\varphi; S)$, as discussed in Section 9.

Unlike our *formulas*, which can take on arbitrary values in $[0,1]$, our *sentences* take on only the values True and False. We refer to our sentences as *multidimensional sentences*, or for short *MD-sentences*.* For a fixed k, we refer to the MD-sentence $(\sigma_1, \ldots, \sigma_k; S)$ as k*-dimensional*. The class of MD-sentences is robust. In particular, Theorem 4.2 says that each finite Boolean combination of MD-sentences is equivalent to

*Note that we are not saying that the *logic* is multidimensional (which could mean that the values taken on by variables are vectors, not just numbers), but instead we are saying that the *sentences* in our "outer" logic are multidimensional. The "inner" logic we work with in this paper is real-valued, and real-valued logic has been heavily studied. What is novel in our paper are our multidimensional sentences.

a single MD-sentence. We give a sound and (finite-strongly) complete axiomatization that is parameterized to deal simultaneouly with many real-valued logics. This axiomatization allows us to derive exactly what information can be inferred about the combinations of truth values of a collection of formulas given information about the combinations of truth values of other collections of formulas.

Given a model M and a sentence $\gamma = (\sigma_1, \ldots, \sigma_k; S)$, we now say what it means for M to satisfy γ. If the value in M of σ_i is s_i (as defined above) for $1 \leq I \leq k$, and if $(s_1, \ldots, s_k) \in S$, then we say that M *satisfies* (or *is a model of*) γ, written $M \vDash \gamma$. Note that if γ is satisfiable, i.e., if γ has some model M, then $S \neq \emptyset$.

2. Axioms and inference rules

We now give our axiom and inference rules. Each of our rules is of the form "from A infer B" or "from A infer B where ...". We refer to A as the *premise* and B as the *conclusion*.

1. We have only one axiom: $(\sigma; [0,1])$. Axiom 1 guarantees that all values are in $[0,1]$.
2. Our first inference rule is: if π is a permutation of $1, \ldots, k$, then from $(\sigma_1, \ldots, \sigma_k; S)$ infer $(\sigma_{\pi(1)}, \ldots, \sigma_{\pi(k)}; S')$, where $S' = \{(s_{\pi(1)}, \ldots, s_{\pi(k)}) \colon (s_1, \ldots, s_k) \in S\}$. Rule 2 simply permutes the order of the components.
3. Our next inference rule is: from $(\sigma_1, \ldots, \sigma_k; S)$ infer $(\sigma_1, \ldots, \sigma_k, \sigma_{k+1}, \ldots, \sigma_m; S \times [0,1]^{m-k})$. Rule 3 extends $(\sigma_1, \ldots, \sigma_k; S)$ to include $\sigma_{k+1}, \ldots, \sigma_m$ with no nontrivial information being given about the new components.
4. Our next inference rule is: from $(\sigma_1, \ldots, \sigma_k; S_1)$ and $(\sigma_1, \ldots, \sigma_k; S_2)$ infer $(\sigma_1, \ldots, \sigma_k; S_1 \cap S_2)$. Rule 4 enables us to join the information in $(\sigma_1, \ldots, \sigma_k; S_1)$ and $(\sigma_1, \ldots, \sigma_k; S_2)$.
5. Our next inference rule is the following (where $0 < r < k$): from $(\sigma_1, \ldots, \sigma_k; S)$ infer $(\sigma_1, \ldots, \sigma_{k-r}; S')$, where $S' = \{(s_1, \ldots, s_{k-r}) \colon (s_1, \ldots, s_k) \in S\}$. Intuitively, S' is the projection of S onto the first $k-r$ components. Rule 5 enables us to select information about $\sigma_1, \ldots, \sigma_{k-r}$ from information about $\sigma_1, \ldots, \sigma_k$.
6. Our next inference rule is: from $(\sigma_1, \ldots, \sigma_k; S)$ infer $(\sigma_1, \ldots, \sigma_k; S')$ if $S \subseteq S'$. Rule 6 says that we can go from more information to less information. The intuition is that smaller information sets are more informative.

We now give an inference rule that depends on the real-valued logic under consideration. For each connective α, let f_α be as defined in Section 1. In the sentence $(\sigma_1, \ldots, \sigma_k; S)$, let us say that the tuple $(s_1, \ldots, s_k)$ in S is *good* if (a) $s_m = f_\alpha(s_i, s_j)$ whenever σ_m is $\sigma_i \, \alpha \, \sigma_j$ and α is a binary connective (such as &), and (b) $s_j = f_\neg(s_i)$ whenever σ_j is $\neg\sigma_i$. Note that being "good" is a local property of a tuple s in S (that is, it depends only on the tuple s and not on the other tuples in S). Of course, if the real-valued logic under consideration has other connectives, possibly of higher arity, then we would modify the definition of a good tuple in the obvious way.

7. We then have the following inference rule: from $(\sigma_1, \ldots, \sigma_k; S)$ infer $(\sigma_1, \ldots, \sigma_k; S')$ when S' is the set of good tuples of S. Rule 7 is our key rule of inference.

   Let γ_1 be the premise $(\sigma_1, \ldots, \sigma_k; S)$ and let γ_2 be the conclusion $(\sigma_1, \ldots, \sigma_k; S')$ of Rule 7. As we shall discuss later, γ_1 and γ_2 are logically equivalent (that is, every model of one is a model of the other), and S' is as small as possible so that γ_1 and γ_2 are logically equivalent.

A simple example of a valid sentence (that is, a sentence rue in every model) is $(A, B, A \veebar B; S)$ where $S = \{(s_1, s_2, s_3)\colon \ s_1 \in [0,1], s_2 \in [0,1], s_3 = f_{\veebar}(s_1, s_2)\}$. This is derived from the valid sentence $(A, B, A \veebar B; [0,1]^3)$ by applying Rule 7.

3. Soundness and completeness of MD-sentences

We need the notion of closure under subformulas. If α is a binary connective, then the *subformulas* of $\sigma_1 \, \alpha \, \sigma_2$ are σ_1 and σ_2. The subformula of $\neg\sigma$ is σ. Let Γ be a set of MD-sentences. We define the *closure G of Γ under subformulas* as follows. For each sentence $(\gamma_1, \ldots, \gamma_m; S)$ in Γ, the set G contains $\gamma_1, \ldots, \gamma_m$, and for each formula γ in G, the set G contains every subformula of γ. In particular, G contains every atomic proposition that appears inside the components of Γ.

Let Γ be a finite set of MD-sentences, and let γ be a single MD-sentence. We write $\Gamma \vDash \gamma$ if every model of Γ is a model of γ. We write $\Gamma \vdash \gamma$ if there is a proof of γ from Γ, using our axiom system. *Soundness* says "$\Gamma \vdash \gamma$ implies $\Gamma \vDash \gamma$". *Completeness* says "$\Gamma \vDash \gamma$ implies $\Gamma \vdash \gamma$" (earlier, we referred to this notion as "finite-strongly completeness"). In this section, we shall prove that our axiom system is sound and complete for MD-sentences.

We now explain why it is necessary, in the case of Łukasiewicz logic, to assume that Γ is finite in the definition of completeness. (In our explanation, we make use of ideas from (13).) Let A^k denote $A \,\&\, A \,\&\, \cdots \,\&\, A$, where A appears k times. Let Γ be the infinite set of sentences containing $(A; [0,1))$ and $(B \to A^k; \{1\})$ for each integer $k \geq 1$. Thus, Γ says that the value of A is strictly less than 1 and that $B \to A^k$ takes on the value 1 for each $k \geq 1$. Let γ be $(B; \{0\})$, which says that B takes on the value 0. We now show that Γ logically implies γ. Assume not. Then there is a model M where Γ holds but γ does not, and so B does not take the value 0. In this model M, since Γ holds, the value of A is less than 1. It then follows from the definition of conjunction in Łukasiewicz logic that in the model M, there is k such that A^k has value 0. From $(B \to A^k; \{1\})$ this then implies that in the model M, the value of B is 0, a contradiction. Hence, Γ logically implies γ. Because our proofs are of finite length, there cannot be a proof of γ from Γ, since this would give a proof of γ from a finite subset of Γ, but no finite subset of Γ logically implies τ. In the case of Gödel logic, it is all right for Γ to be infnite, since Gödel logic satisfies a compactness theorem, which says that if $\Gamma \vDash \gamma$, then there is a finite subset Γ' of Γ such that $\Gamma' \vDash \gamma$ (22).

We define a special property of certain MD-sentences, that is used in a crucial manner in our completeness proof. Let us say that a sentence $(\sigma_1, \ldots, \sigma_k; S)$ is *minimized* if whenever $(s_1, \ldots, s_k) \in S$, then there is a model M of $(\sigma_1, \ldots, \sigma_k; S)$ such that for $1 \leq i \leq k$, the value of σ_i in M is s_i. Thus, $(s_1, \ldots, s_k) \in S$ *if and only if* there is a model M of $(\sigma_1, \ldots, \sigma_k; S)$ such that for $1 \leq i \leq k$, the value of σ_i in M is s_i. We use the word "minimized", since intuitively, S is as small as possible. Note that there can be no algorithm for deciding if an MD-sentence is minimized, since there are uncountably many MD-sentences (because there are uncountably many choices for S).

Our completeness proof makes use of the following lemmas.

Lemma 3.1 *Let $(\sigma_1, \dots, \sigma_k; S)$ be the premise of Rule 7. Assume that $G = \{\sigma_1, \dots, \sigma_k\}$ is closed under subformulas (so that in particular, every atomic proposition that appears inside a member of G is a member of G). Then the conclusion $(\sigma_1, \dots, \sigma_k; S')$ of Rule 7 is minimized.*

Proof Let φ be the conclusion $(\sigma_1, \dots, \sigma_k; S')$ of Rule 7. Assume that $(s_1, \dots, s_k) \in S'$. To prove that φ is minimized, we must show that there is a model M of φ such that for $1 \le i \le k$, the value of σ_i in M is s_i. From the assignment of values to the atomic propositions, as specified by a portion of $(s_1, \dots, s_k)$, we obtain our model M. For this model M, the value of each σ_i is exactly that specified by $(s_1, \dots, s_k)$, as we can see by a simple induction on the structure of formulas. Hence, φ is minimized. □

The assumption of closure under subformulas in Lemma 3.1 is needed, as the following example shows. Let γ be the MD-sentence $(\sigma_1 \,\&\, \sigma_2, \sigma_1 \veebar \sigma_2; \{(0.5, 0.2)\})$ in Gödel logic. The result of applying Rule 7 to γ is γ itself because neither of its components include the other as a subformula. But γ is not minimized, since it is not satisfiable, because the min of two numbers cannot be greater than the max.

Lemma 3.2 *For each of Rules 2, 3, and 7, the premise is logically equivalent to the conclusion. For Rule 4, the set of the premises is logically equivalent to the conclusion.*

Proof The equivalence of the premise and conclusion of Rule 2 is clear. For Rules 3 and 7, the fact that the premise logically implies the conclusion follows from soundness of the rules, as does the fact that the set of the premises of Rule 4 logically implies the conclusion, and we shall show soundness shortly. We now show that for Rules 3 and 7, the conclusion logically implies the premise. For Rule 3, we see that if $(s_1, \dots, s_m) \in S \times [0,1]^{m-k}$, then $(s_1, \dots, s_k) \in S$. Hence, the conclusion of Rule 3 logically implies the premise of Rule 3. For Rule 7, the conclusion logically implies the premise because of the soundness of Rule 6. For Rule 4, the conclusion logically implies the each of the premises, and hence the set of the premises, because of the soundness of Rule 6. □

Lemma 3.3 *Minimization is preserved by Rules 2 and 4, in the following sense.*

1. *If the premise of Rule 2 is minimized, then so is the conclusion.*
2. *If the premises $(\sigma_1, \dots, \sigma_k; S_1)$ and $(\sigma_1, \dots, \sigma_k; S_2)$ of Rule 4 are minimized, then so is the conclusion $(\sigma_1, \dots, \sigma_k; S_1 \cap S_2)$.*

Proof Part (1) is immediate, since the premise and conclusion have exactly the same information.

For part (2), assume that $(\sigma_1, \dots, \sigma_k; S_1)$ and $(\sigma_1, \dots, \sigma_k; S_2)$ are minimized. To show that $(\sigma_1, \dots, \sigma_k; S_1 \cap S_2)$ is minimized, we must show that if $(s_1, \dots, s_k) \in S_1 \cap S_2$, then there is a model M of $(\sigma_1, \dots, \sigma_k; S_1 \cap S_2)$ such that for $1 \le i \le k$, the value of σ_i in M is s_i. Assume that $(s_1, \dots, s_k) \in S_1 \cap S_2$. Hence, $(s_1, \dots, s_k) \in S_1$. Since $(\sigma_1, \dots, \sigma_k; S_1)$ is minimized, we obtain the desired model M. □

Theorem 3.4 *Our axiom system is sound and complete for MD-sentences.*

Proof We begin by proving soundness. We say that an axiom is sound if it is true in every model. We say that an inference rule is sound if every model that satisfies the premise also satisfies the conclusion. To prove soundness of our axiom system, it is sufficient to show that our axiom is sound and that each of our rules is sound.

Axiom 1 is sound, since every real-valued logic formula has a value in $[0, 1]$.

Rule 2 is sound, since the premise and conclusion encode exactly the same information.

Rule 3 is sound for the following reason. Let M be a model, and let $s_1, \dots, s_m$ be the values of $\sigma_1, \dots, \sigma_m$, respectively, in M. If M satisfies the premise, then $(s_1, \dots, s_k) \in S$. This implies that $(s_1, \dots, s_m) \in S \times [0,1]^{m-k})$ and so M satisfies the conclusion.

Rule 4 is sound for the following reason. Let M be a model, and let $s_1, \dots, s_k$ be the values of $\sigma_1, \dots, \sigma_k$, respectively, in M. If M satisfies the premise, then $(s_1, \dots, s_k) \in S_1$ and $(s_1, \dots, s_k) \in S_2$. Therefore, $(s_1, \dots, s_k) \in S_1 \cap S_2$, and so M satisfies the conclusion.

Rule 5 is sound for the following reason. Let M be a model, and let $s_1, \dots, s_k$ be the values of $\sigma_1, \dots, \sigma_k$, respectively, in M. If M satisfies the premise, then $(s_1, \dots, s_k) \in S$. Therefore $(s_1, \dots, s_{k-r}) \in S'$, and so M satisfies the conclusion.

Rule 6 is sound for the following reason. Let M be a model, and let $s_1, \dots, s_k$ be the values of $\sigma_1, \dots, \sigma_k$, respectively, in M. If M satisfies the premise, then $(s_1, \dots, s_k) \in S$. Therefore, $(s_1, \dots, s_k) \in S'$, and so M satisfies the conclusion.

Rule 7 is sound for the following reason. Let M be a model, and let $s_1, \dots, s_k$ be the values of $\sigma_1, \dots, \sigma_k$, respectively, in M. If M satisfies the premise, then $(s_1, \dots, s_k) \in S$. In our real-valued logic, we have that (a) $f_\alpha(s_i, s_j) = s_m$ when σ_m is $\sigma_i \, \alpha \, \sigma_j$ and α is a binary connective (such as &), and (b) $f_\neg(s_i) = s_j$ when σ_j is $\neg\sigma_i$. So the tuple $(s_1, \dots, s_k)$ is good, and hence in S', so M satisfies the conclusion.

This completes the proof of soundness. We now prove completeness. Assume that Γ is finite, and $\Gamma \models \gamma$; we must show that $\Gamma \vdash \gamma$. We can assume without loss of generality that Γ is nonempty, because if Γ is empty, we replace it by a singleton set containing an instance of our Axiom 1.

Let $\Gamma = \{\gamma_1, \dots, \gamma_n\}$. For $1 \le i \le n$, assume that γ_i is $(\sigma^i_1, \dots, \sigma^i_{k_i}; S_i)$, and let $\Gamma_i = \{\sigma^i_1, \dots, \sigma^i_{k_i}\}$. Assume that γ is $(\sigma^0_1, \dots, \sigma^0_{k_0}; S_0)$, and let $\Gamma_0 = \{\sigma^0_1, \dots, \sigma^0_{k_0}\}$. Let G be the closure of $\Gamma_0 \cup \Gamma_1 \cup \dots \cup \Gamma_n$ under subformulas.

For each i with $1 \le i \le n$, let H_i be the set difference $G \setminus \Gamma_i$. Let $r_i = |H_i|$. Let $H_i = \{\tau^i_1, \dots \tau^i_{r_i}\}$. By applying Rule 3, we prove from γ_i the sentence $(\sigma^i_1, \dots, \sigma^i_{k_i}, \tau^i_1, \dots, \tau^i_{r_i}; S_i \times [0,1]^{r_i})$. Let ψ_i be the conclusion of Rule 7 when the premise is $(\sigma^i_1, \dots, \sigma^i_{k_i}, \tau^i_1, \dots, \tau^i_{r_i}; S_i \times [0,1]^{r_i})$.

Let $\delta_1, \dots, \delta_p$ be a fixed ordering of the members of G. Since the set of components of each ψ_i is G, we can use Rule 2 to rewrite ψ_i as a sentence $(\delta_1, \dots, \delta_p; T_i)$. Let us call this sentence φ_i.

Also, since the only rules used in proving φ_i from γ_i are Rules 2, 3, and 7, it follows from Lemma 3.2 that γ_i and φ_i are logically equivalent.

We now make use of the notion of minimization. Let $T = T_1 \cap \cdots \cap T_n$. Define φ to be the sentence $(\delta_1, \ldots, \delta_p; T)$. It follows from Lemma 3.1 that each ψ_i is minimized. So by Lemma 3.3, each φ_i is minimized. By Lemma 3.3 again, φ is minimized.

The sentence φ was obtained from the sentences φ_i by applying Rule 4 $n-1$ times. It follows from Lemma 3.2 that φ is equivalent to $\{\varphi_1, \ldots, \varphi_n\}$. Since we also showed that γ_i is logically equivalent to φ_i for $1 \leq i \leq n$, it follows that φ is logically equivalent to Γ. Hence, since $\Gamma \vDash \gamma$, it follows that $\{\varphi\} \vDash \gamma$. It also follows that to prove that $\Gamma \vdash \gamma$, we need only show that there is a proof of γ from φ.

Recall that γ is $(\sigma_1^0, \ldots, \sigma_{k_0}^0; S_0)$, and φ is $(\delta_1, \ldots, \delta_p; T)$. By applying Rule 2, we can re-order the components of φ so that the components start with $\sigma_1^0, \ldots, \sigma_{k_0}^0$. We thereby obtain from φ a sentence $(\sigma_1^0, \ldots, \sigma_{k_0}^0, \ldots; T')$, which we denote by φ'. By Lemma 3.2 we know that φ and φ' are logically equivalent. So $\{\varphi'\} \vDash \gamma$. Since φ is minimized, so is φ', by Lemma 3.3. By applying Rule 5, we obtain from φ' a sentence $(\sigma_1^0, \ldots, \sigma_{k_0}^0; T'')$, which we denote by φ''.

We now show that $T'' \subseteq S_0$. This is sufficient to complete the proof of completeness, since then we can use Rule 6 to prove γ. If T'' is empty, we are done. So assume that $(s_1, \ldots, s_{k_0}) \in T''$; we must show that $(s_1, \ldots, s_{k_0}) \in S_0$.

Since $(s_1, \ldots, s_{k_0}) \in T''$, it follows that there is an extension $(s_1, \ldots, s_{k_0}, \ldots, s_p)$ in T'. Since φ' is minimized, there is a model M of φ' such that the value of σ_i^0 is s_i, for $1 \leq i \leq k_0$. Since $\{\varphi'\} \vDash \gamma$, it follows that M is a model of γ. By definition of what it means for M to be a model of γ, it follows that $(s_1, \ldots, s_{k_0}) \in S_0$, as desired.

This completes the soundness and completeness proofs. □

4. Boolean combinations of MD-sentences

Our main theorem in this section implies that MD-sentences are robust, in that each finite Boolean combination of MD-sentences is eqvuivealnt to a single MD-sentence. Of course, since we are dealing with sentences (which take only the values True and False) in our "outer" logic, we use the standard Boolean connectives. Shortly, we shall make these notions precise.

In this section, there will be two disjoint sets of atomic propositions. The first are the atomic propositions appearing inside MD-sentences; we call these *MD-atomic propositions*. For example, in the MD-sentence $(A \,\&\, B, A \veebar B; [0.3, 0.7] \times [0.5, 1])$, the MD-atomic propositions are A and B. The second are those atomic propositions appearing inside propositional formulas; we call these *prop-atomic propositions*. For example, in the propositional formula $X \vee (\neg X \wedge Y)$, the prop-atomic propositions are X and Y.

We now define *extended MD-sentences*. Let γ be a propositional formula (built using $\wedge$, $\vee$, and $\neg$), and let f be a function mapping each prop-atomic proposition appearing in γ to an MD-sentence. Then the result of replacing each prop-atomic proposition X in γ by $f(X)$ is an extended MD-sentence. For example, let γ be the propositional formula $X \vee (\neg X \wedge Y)$, let $f(X) = (\sigma_1; S)$, and let $f(Y) = (\sigma_1', \sigma_2'; S')$. We then get the extended MD-sentence $(\sigma_1; S) \vee (\neg(\sigma_1; S) \wedge (\sigma_1', \sigma_2'; S'))$.

This defines the syntax of extended MD-sentences. We now define their semantics. As before, a model M is an assignment g^M of truth values to the MD-atomic propositions. Let γ be a propositional formula (built using $\wedge$, $\vee$, and $\neg$), and let f be a function mapping each prop-atomic proposition appearing in γ to an MD-sentence. Let the result of replacing each prop-atomic proposition X in γ by $f(X)$ be the extended MD-sentence γ'. We now say what it means for the model M to model, or satisfy, γ'. For each prop-atomic proposition X appearing in γ, let $f'(X) =$ True if $M \vDash f(X)$, and otherwise let $f'(X) =$ False. Now let γ'' be the result of replacing every prop-atomic proposition X in γ by $f'(X)$. The result is logically equivalent to either True or False. If this result is logically equivalent to True, then we say that M models γ', written $M \vDash \gamma'$. Let us consider our example above, where γ is the propositional formula $X \vee (\neg X \wedge Y)$, and $f(X) = (\sigma_1; S)$, and $f(Y) = (\sigma_1', \sigma_2'; S')$. This gives the extended MD-sentence γ', which is $(\sigma_1; S) \vee (\neg(\sigma_1; S) \wedge (\sigma_1', \sigma_2'; S'))$. If $M \nvDash (\sigma_1; S)$ but $M \vDash (\sigma_1', \sigma_2'; S')$, then γ'' is False $\vee$ ($\neg$False $\wedge$ True), which is logically equivalent to True. So $M \vDash \gamma'$.

Theorem 4.1 *Every extended MD-sentence is logically equivalent to a single MD-sentence.*

Proof Let γ be a propositional formula built using $\wedge$, $\vee$, and $\neg$. Assume that the extended MD-sentence γ' is obtained from γ by replacing each prop-atomic proposition in γ with an MD-sentence.

We prove the theorem by induction on the structure of γ', working from the inside out. Thus, we show (a) if τ_1 and τ_2 are MD-sentences, then the extended MD-sentence $\tau_1 \vee \tau_2$ is logically equivalent to an MD-sentence; (b) if τ_1 and τ_2 are MD-sentences, then the extended MD-sentence $\tau_1 \wedge \tau_2$ is logically equivalent to an MD-sentence; and (c) if τ_1 is an MD-sentence, then the extended MD-sentence $\neg\tau_1$ is logically equivalent to an MD-sentence. Let τ_1 and τ_2 be MD-sentences. Assume that τ_1 is $(\sigma_1^1, \ldots, \sigma_m^1; S_1)$, and that τ_2 is $(\sigma_1^2, \ldots, \sigma_n^2; S_2)$. As in the proof of Theorem 3.4, let G be the closure of $\{\sigma_1^1, \ldots, \sigma_m^1, \sigma_1^2, \ldots, \sigma_n^2\}$ under subformulas. Assume that $G = \{\delta_1, \ldots, \delta_p\}$. As in the proof of Theorem 3.4, we know that for $i = 1$ and $i = 2$, there is T_i such that τ_i is equivalent to a sentence $(\delta_1, \ldots, \delta_p; T_i)$. We now show that the disjunction $\tau_1 \vee \tau_2$ is equivalent to $(\delta_1, \ldots, \delta_p; T_1 \cup T_2)$. Let M be a model, and assume that the value of δ_i in M is s_i, for $1 \leq I \leq p$. If M satisfies $\tau_1 \vee \tau_2$, then $(s_1, \ldots, s_p) \in T_1$ or $(s_1, \ldots, s_p) \in T_2$. Hence, $(s_1, \ldots, s_p) \in T_1 \cup T_2$, so M satisfies $(\delta_1, \ldots, \delta_p; T_1 \cup T_2)$. Conversely, if M satisfies $(\delta_1, \ldots, \delta_p; T_1 \cup T_2)$, then $(s_1, \ldots, s_p) \in T_1 \cup T_2$, and hence either $(s_1, \ldots, s_p) \in T_1$, in which case M satisfies τ_1, or $(s_1, \ldots, s_p) \in T_2$, in which case M satisfies τ_2. Therefore, M satisfies $\tau_1 \vee \tau_2$,as desired. A similar argument shows that the conjunction $\tau_1 \wedge \tau_2$ is equivalent to $(\delta_1, \ldots, \delta_p; T_1 \cap T_2)$, and the negation $\neg\gamma_1$ is equivalent to $(\delta_1, \ldots, \delta_p; \tilde{T}_i)$, where $\tilde{T}_1$ is the set difference $[0,1]^p \setminus T_1$. □

A good way to view Theorem 4.1 is as follows:

Theorem 4.2 *Each finite Boolean combination of MD-sentences is equivalent to a single MD-sentence.*

Proof This is really just a restating of Theorem 4.1. □

5. Reducing the dimensionality

In this section, we give both a negative and a positive result about reducing the dimensionality of MD-sentences. We then give an open problem.

Theorem 5.1 *There is a 2-dimensional MD-sentence that is not equivalent (in either Łukasiewicz or Gödel logic) to a 1-dimensional MD-sentence.*

Proof Let σ be the 2-dimensional MD-sentence $(A_1, A_2; S)$ where $S = \{(a_1, a_2) : a_1^2 = a_2\}$. We now show that σ is not equivalent to a 1-dimensional MD-sentence. If φ is a formula in our set F of logical formulas, and φ involves only A_1 and A_2, then it is easy to see (by induction on the structure of formulas) that for Łukasiewicz or Gödel logic, φ defines a piecewise linear function g_φ, in the sense that the 1-dimensional MD-sentence $(\varphi; S')$ says that if a_1 is the value of A_1 and a_2 is the value of A_2, then $g_\varphi(a_1, a_2) \in S'$. Since there is no such piecewise linear function g_φ and set S' for our sentence σ, the result holds. □

The next theorem does not depend on restricting to Łukasiewicz or Gödel logic.

Theorem 5.2 *Every finite set of MD-sentences of arbitrary dimensions that involve only the k atomic propositions $A_1, \ldots, A_k$ is equivalent to a single k-dimensional MD sentence $(A_1, \ldots, A_k; S)$. (The set S depends on the real-valued logic being considered.)*

Proof Let Γ be a finite set of MD-sentences. We can view Γ as a conjunction of MD-sentences, so by Theorem 4.1, Γ is equivalent to a single MD-sentence γ. As in the proof of completeness, by closing under subformulas, applying Rule 7, and reordering by applying Rules 2, we obtain an MD-sentence $(A_1, \ldots, A_k, \varphi_1, \ldots, \varphi'_r; S')$ that is equivalent to γ. Since the tuples in S' are good tuples, this is equivalent to the sentence $(A_1, \ldots, A_k; S)$ where $S = \{(s_1, \ldots, s_k) : (s_1, \ldots s_k, s'_1, \ldots s'_r) \in S'\}$. □

Open problem: For each k with $k \geq 2$, does there exist a $(k+1)$-dimensional MD-sentence that in Łukasiewicz or Gödel logic is not equivalent to a k-dimensional MD-sentence?

6. SoCRAtic: A decision procedure

Given a finite set Γ of MD-sentences, and a single MD-sentence γ, Theorem 3.4 says that $\Gamma \vDash \gamma$ if and only if $\Gamma \vdash \gamma$. As we shall show, under natural assumptions there is an algorithm for deciding if $\Gamma \vDash \gamma$. We call this algorithm a *decision procedure.* If the information sets S all have s simple structure and the size of Γ is treated as a constant, than the algorithm runs in polynomial time.

It is natural to wonder whether we can simply use our complete axiomatization to derive a decision procedure. The usual answer is that it is not clear in what order to apply the rules of inference. In our proof of completeness, the rules of inference are applied in a specific order, so that is not an issue here. Rather, the problem is that in applying Rule 7, there is no easy way to derive S' from S, even if S is fairly simple. In fact, we now show that even deciding if S' is nonempty is NP-hard. Let φ be an instance of the NP-hard problem 3SAT. Thus, φ is of the form $(B_1^1 \veebar B_2^1 \veebar B_3^1) \& \cdots \& (B_1^r \veebar B_2^r \veebar B_3^r)$, where each B_j^i is a literal (an atomic proposition or its negation). Assume that the atomic propositions that appear in φ are $A_1, \ldots, A_k$. Let ψ be the sentence

$$(A_1, \ldots, A_k, \neg A_1, \ldots, \neg A_k, \tau_1, \ldots, \tau_r, \tau_1 \veebar B_3^1, \ldots, \tau_r \veebar B_3^r; S),$$

where τ_i is $B_1^i \veebar B_2^i$, for $1 \leq i \leq r$, and where $S = \{0,1\}^{2k+r} \times \{1\}^r$. Assume that we apply Rule 7 where the premise is ψ, and the conclusion is

$$(A_1, \ldots, A_k, \neg A_1, \ldots, \neg A_k, \tau_1, \ldots, \tau_r, \tau_1 \veebar B_3^1, \ldots, \tau_r \veebar B_3^r; S').$$

We call this sentence ψ'. It follows easily from our construction of ψ that the 3SAT problem φ is satisfiable if and only if ψ is satisfiable. Now ψ and ψ' are logically equivalent, by Lemma 3.2. So the 3SAT problem φ is satisfiable if and only if ψ' is satisfiable. By Lemma 3.1, we know that ψ' is minimized. Hence, if S' is nonempty, there is a model of ψ', by the definition of minimization. And if S' is empty, then by the definition of a model of a sentence, there is no model of ψ'. Therefore, ψ' is satisfiable if and only if S' is nonempty. By combining this with our earlier observation that the 3SAT problem φ is satisfiable if and only if ψ' is satisfiable, it follows that the 3SAT problem φ is satisfiable if and only if S' is nonempty. Hence, deciding if S' is nonempty is NP-hard.

We now discuss our decision procedure, which bears resemblance to Reiner Hähnle's decision procedure for the tableaux method with infinite-valued Łukasiewicz logic (23) but extends support to discontinuous operators. Our decision procedure makes use of linear programming and is thus particularly suited for Łukasiewicz and Gödel logic's piecewise linear connective functions; we focus primarily on these two logics in the following, however it is also possible for our decision procedure to work on product logic using the same logarithmic transform as in (24). To have a chance of there being a decision procedure, the set portion S of an MD-sentence $(\sigma_1, \ldots, \sigma_k; S)$ must be tractable. We now give a simple, natural choice for the set portions. A *rational interval* is a subset of $[0, 1]$ that is of one of the four forms (a, b), $[a, b]$, $(a, b]$, or $[a, b)$, where a and b are rational numbers. Let us say that a sentence $(\sigma_1, \ldots, \sigma_k; S)$ is *interval-based* if S is of the form $S_1 \times \cdots \times S_k$, where each S_i is a union of a finite number of rational intervals. If each S_i is the union of at most N rational intervals, then we say that the sentence is *N-interval-based.* Note that this interval-based sentence $(\sigma_1, \ldots, \sigma_k; S)$ is equivalent to the set $\{(\sigma_1; S_1), \ldots, (\sigma_k; S_k)\}$ of 1-dimensional sentences. This observation is useful in implementing the decision procedure.

Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$. For $1 \leq i \leq n$, assume that γ_i is $(\sigma_1^i, \ldots, \sigma_{k_i}^i; S_i)$, and let $\Gamma_i = \{\sigma_1^i, \ldots, \sigma_{k_i}^i\}$. Assume that γ is $(\sigma_1^0, \ldots, \sigma_{k_0}^0; S_0)$, and let $\Gamma_0 = \{\sigma_1^0, \ldots, \sigma_{k_0}^0\}$. Let G be the closure of $\Gamma_0 \cup \Gamma_1 \cup \cdots \cup \Gamma_n$ under subformulas. If $|G| \leq M$, then we say that the pair (Γ, γ) *has nesting depth at most M.*

Theorem 6.1 *Assume either Łukasiewicz logic or Gödel logic, with the connectives $\&$, $\veebar$, $\rightarrow$, and $\neg$. Assume that $\Gamma \cup \{\gamma\}$ is interval based. Then there is an algorithm that determines whether $\Gamma \vDash \gamma$. Assume that Γ has at most P sentences, each sentence in $\Gamma \cup \{\gamma\}$ is N-interval based, and (Γ, γ) has nesting depth at most M. If M is fixed, then the algorithm runs in time polynomial in P and N.*

Proof Assume throughout the proof that Γ has at most P sentences, each sentence in $\Gamma \cup \{\gamma\}$ is N-interval based, and (Γ, γ) has nesting depth at most M.

It is easy to see that $\Gamma \vDash \gamma$ if and only $\Gamma \cup \{\neg\gamma\}$ is not satisfiable. So we need only give an algorithm that decides whether $\Gamma \cup \{\neg\gamma\}$ is satisfiable.

Let $\{\sigma_1, \ldots, \sigma_p\}$ be the closure of $\Gamma \cup \{\gamma\}$ under subformulas. Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$. By making use of Rules 2 and 3, for each i with $1 \le i \le n$, we can create a sentence γ'_i of the form $(\sigma_1, \ldots, \sigma_p; S^i)$ that by Lemma 3.2 is equivalent to γ_i, and that has $\sigma_1, \ldots, \sigma_p$ as components. By the construction, each γ'_i is N-interval-based.

Similarly, create the sentence γ' of the form $(\sigma_1, \ldots, \sigma_p; T)$ that is equivalent to γ, and that has $\sigma_1, \ldots, \sigma_p$ as components. As before, γ' is N-interval-based.

Now Γ is equivalent to the conjunction of the sentences γ'_i for $1 \le i \le n$, and this conjunction is equivalent to $(\sigma_1, \ldots, \sigma_p; S)$, where $S = \bigcap_{i \le n} S^i$. We now show that $(\sigma_1, \ldots, \sigma_p; S)$ is PN-interval-based. By assumption, for each i with $1 \le i \le n$, we have that S^i is of the form $S^i_1 \times \cdots \times S^i_p$, where each S^i_j is the union of at most N intervals. For each j with $1 \le j \le p$, let $S_j = \bigcap_i S^i_j$. Then $S = S_1 \times \cdots \times S_p$. So to show that $(\sigma_1, \ldots, \sigma_p; S)$ is PN-interval-based, we need only show that each S_j is the union of at most PN intervals.

Since $S_j = \bigcap_{i \le n} S^i_j$, where each S^i_j is the union of at most N intervals, we see that S_j is the union of intervals where the left endpoint of each interval in S_j is one of the left endpoints of intervals in $\bigcup_{i \le n} S^i_j$. For each j, there are n sets S^i_j. And for each i with $1 \le i \le n$, there are at most N left endpoints of S^i_j. So the total number of left endpoints of intervals in $\bigcup_{i \le n} S^i_j$ is at most $nN \le PN$, and so the number of intervals in S_j is at most PN. Since $S = S_1 \times \cdots \times S_p$, it follows that $(\sigma_1, \ldots, \sigma_p; S)$ is PN-interval-based.

Let us now consider $\neg\gamma$, which is equivalent to $\neg\gamma'$. Recall that γ' is $(\sigma_1, \ldots, \sigma_p; T)$, and that γ' is N-interval-based. So T is of the form $T_1 \times \cdot \times T_p$, where each T_j is the union of at most N intervals. As discussed earlier, the negation of γ' is $(\sigma_1, \ldots, \sigma_p; \tilde{T})$, where $\tilde{T}$ is the set difference $[0,1]^p \backslash T$. For each j with $1 \le j \le p$, let T'_j be the set difference $[0,1] \backslash T_j$. Clearly, T'_j is the union of intervals. The left endpoints of intervals in T'_j are the right-end points of intervals in T_j, possible along with 0. So T'_j is the union of at most $N+1$ intervals. Let $V_j = [0,1]^{j-1} \times T'_j \times [0,1]^{p-j}$. It is straightforward to see that $\tilde{T} = \bigcup_{j \le p} V_j$.

Now, showing that $\Gamma \cup \{\neg\gamma\}$ is not satisfiable is equivalent to showing that $(\sigma_1, \ldots, \sigma_p; S) \wedge (\sigma_1, \ldots, \sigma_p; \tilde{T})$ is not satisfiable, which is equivalent to showing that for every j with $1 \le j \le p$, we have that $(\sigma_1, \ldots, \sigma_p; S) \wedge (\sigma_1, \ldots, \sigma_p; V_j)$ is not satisfiable. So we need only give an algorithm for deciding if $(\sigma_1, \ldots, \sigma_p; S) \wedge (\sigma_1, \ldots, \sigma_p; V_j)$ is satisfiable. Let us hold j fixed. Since, as we showed, $(\sigma_1, \ldots, \sigma_p; S)$ is PN-interval-based, we can write S as $S_1 \times \cdots \times S_p$, where each S_i is the union of at most PN intervals. Now $(\sigma_1, \ldots, \sigma_p; S) \wedge (\sigma_1, \ldots, \sigma_p; V_j)$ is equivalent to $(\sigma_1, \ldots, \sigma_p; S \cap V_j)$. Now $S \cap V_j$ is of the form $S'_1 \times \cdots \times S'_p$, where $S'_m = S_m$ for $m \ne j$, and where $S'_j = S_j \cap T'_j$. We showed that T'_j is the union of at most $N+1$ intervals, and that S_j is the union of at most PN intervals, so it follows that $S_j \cap T'_j$ is the union of at most $PN + N + 1$ intervals, since each left endpoint of the intervals in $S_j \cap T'_j$ is a left endpoint of an interval in S_j or an interval in T'_j.

We now describe our algorithm for deciding if the sentence $(\sigma_1, \ldots, \sigma_p; S \cap V_j)$, that is, for the sentence $(\sigma_1, \ldots, \sigma_p; S'_1 \times \cdots \times S'_p)$, which is $(PN+N+1)$-interval-based, is satisfiable. This can be broken into subproblems, one for each choice $(I_1, \ldots, I_p)$ of a single interval I_k from S'_k for each k with $1 \le k \le p$. This gives a total of at most $(PN+N+1)^M$ subproblems. For each of these subproblems, we wish to decide satisfiability of the system $\{s_1 \in I_1, \ldots, s_p \in I_p\}$ along with (a) the binary constraints $f_\alpha(s_i, s_j) = s_m$ when σ_m is $\sigma_i \,\alpha\, \sigma_j$ and α is a &, $\veebar$, or $\rightarrow$, and (b) $f_\neg(s_i) = s_j$ when σ_j is $\neg\sigma_i$.

The constraints $s_j \in I_j$ are specified by inequalities (for example, if I_j is $(a, b]$ we get the inequalities $a < s_i \le b$). We now show how to deal with the constraints in (a) and (b) above. A canonical example is given by dealing with $f_\&(s_i, s_j) = s_m$ in Gödel logic, which interprets "$f_\&(s_i, s_j) = s_m$" as $\min\{s_i, s_j\} = s_m$. We split the system of constraints into two systems of constraints, one where we replace $\min\{s_i, s_j\} = s_m$ by the two statements "$s_i \le s_j$, $s_i = s_m$" and another where we replace $\min\{s_i, s_j\} = s_m$ by the two statements "$s_j < s_i$, $s_j = s_m$". In Łukasiewicz logic, where $f_\&(s_i, s_j)$ is $\max\{0, s_1 + s_2 - 1\}$, we split the system of constraints into two systems of constraints, one where we replace $\max\{0, s_1 + s_2 - 1\} = s_m$ by the two statements "$s_i + s_j - 1 \ge 0$, $s_i + s_j - 1 = s_m$" and another where we replace $\max\{0, s_1 + s_2 - 1\} = s_m$ by the two statements "$s_i + s_j - 1 < 0$, $s_m = 0$". The same approach works for the other binary connectives. For example, in Gödel logic, where $f_\rightarrow(s_i, s_j)$ is 1 if $s_i \le s_j$ and is s_j otherwise, we would split into two cases, one where we replace $f_\rightarrow(s_i, s_j) = s_m$ by the two statements "$s_i \le s_j$, $s_m = 1$" and another where we replace $f_\rightarrow(s_i, s_j) = s_m$ by the two statements "$s_j > s_i$, $s_m = s_j$". In considering the effect of the constraints in (a) and (b), each of our at most $(PN+N+1)^M$ subproblems splits at most $2^p \le 2^M$ times, giving a grand total of at most $(PN+N+1)^M 2^M$ systems of inequalities that we need to check for feasibility (that is, to see if there is a solution). For each of these systems of inequalities, we can make use a polynomial-time algorithm for linear programming to decide feasibility, where the size of each of these systems is linear in M, and so the running time for each instance of the linear programming algorithm is polynomial in M. Since also the number of systems is at most $(PN+N+1)^M 2^M$, and since$M$ is fixed by assumption, this gives us an overall algorithm for decidability, whose running time is polynoimial in N and P. $\square$

The reason we held the parameter M fixed is that the running time of the algorithm is exponential in M, because there are an exponential number of calls to a linear programming subroutine. The algorithm is polynomial-time if there is a fixed bound on M. Such a bound is necessary, because the problem can be co-NP hard, for the following reason.

Let γ be the sentence $(A, \neg A; [1] \times [1])$. Then γ is not satisfiable. Let Γ consist of the single sentence ψ from the beginning of the section. Then $\Gamma \vDash \gamma$ if and only if ψ is not satisfiable. Now ψ is satisfiable if and only if S' from the beginning of the section is nonempty, which we showed is an NP-hard problem to determine. Since $\Gamma \vDash \gamma$ if and only if ψ is not satisfiable, it follows that deciding if $\Gamma \vDash \gamma$ is co-NP hard.

We now give an implementation of the decision procedure. The decision procedure described in the proof of Theorem 6.1 is available from the `socratic-logic` GitHub repository hosted at https://github.com/IBM/socratic-logic. We implemented the algorithm as a Python package named `socratic`, which requires Python 3.6 or newer and makes use of IBM® ILOG® CPLEX® Optimization Studio V12.10.0 or newer via the `docplex` Python package. It would also be possible to implement this same decision procedure using satisfiability modulo theories (SMT) and solvers such as Z3.

A. Implementation details. The implementation closely adheres to the decision procedure described in the proof of Theorem 6.1, though with a few notable design shortcuts.

Boolean variables. One such shortcut is the use of mixed integer linear programming (MILP) to perform the "spliting" of linear programs into two possible optimization problems, specifically by adding a Boolean variable that determines which of a set of constraints must be active. MILP's exploration of either value for the Boolean variable is then equivalent to repeating linear optimization for either possible set of constraints; no feasible solution exists for any combination of Boolean variables in exactly the case that none of the split linear programs are feasible. In practice, CPLEX has built-in support for min, max, abs, and a handful of other functions, though Boolean variables are also useful for implementing Gödel logic's implication, negation, and equivalence operations as well as selecting the specific intervals a sentence's formula truth values lie within.

Strict inequality. The described decision procedure also occasionally calls for continuous constraints with strict inequality, in particular when dealing with the complements of closed intervals, but also when handling input open intervals or the Gödel implication, $(x \to y) = y$ if $x > y$ else 1. To implement a strict inequality constraint such as $x > y$, we introduce a global gap variable $\delta \in [0, 1]$ to widen the distance between either side of the inequality, e.g., $x \geq y + \delta$, and then maximize δ. If optimization yields an apparently feasible solution but with $\delta = 0$, we regard it as infeasible because at least one strict inequality constraint could not be honored strictly.

1-dimensional sentences. We additionally observe that, for theories restricted to interval-based sentences, it is sufficient to support only sentences containing a single formula and collection of truth value intervals, i.e., 1-dimensional sentences of the form $(\sigma; S)$ for a single formula σ. This is because of the following theorem:

Theorem 6.2 *Interval-based sentence* $s = (\sigma_1, \ldots, \sigma_k; S_1 \times \cdots \times S_k)$ *is equivalent to a collection of 1-dimensional sentences* $s_1, \ldots, s_k$, *where* $s_i = (\sigma_i; S_i)$.

Proof Given interval-based sentence s and 1-dimensional sentences $s_1, \ldots, s_k$ as described, apply Rules 3 and 2 to obtain $s'_1, \ldots, s'_k$ given $s'_i = (\sigma_1, \ldots, \sigma_k; [0, 1]^{i-1} \times S_i \times [0, 1]^{k-i})$. One may then repeatedly apply Rule 4 to compose these exactly into s. Likewise, one may apply Rules 2 and 5 to obtain each s_i directly from s. Hence, the two forms are equivalent. □

B. Experimental results. We tested `socratic` in four different experimental contexts:

- 3SAT and higher k-SAT problems which become satisfiable if any one of their input clauses is removed
- 82 axioms and tautologies taken from Hájek in (13), some of which hold only for one of Łukasiewicz or Gödel logic
- A formula given in Formula 2 that is classically valid but invalid in both Łukasiewicz and Gödel logic unless propositions are constrained to be Boolean
- A stress test on sentences with thousands of intervals

Experiments are conducted on a MacBook Pro with macOS Catalina 10.15.5, 2.9 GHz Quad-Core Intel Core i7, 16 GB 2133 MHz LPDDR3, and Intel HD Graphics 630 1536 MB.

k-SAT. We construct classically unsatisfiable k-SAT problems of the form

$$(x_1 \wedge \neg x_1) \vee \cdots \vee (x_k \wedge \neg x_k) \quad [1]$$

which, after CNF conversion, and replacing $\vee$ by $\veebar$, yields for 3SAT

$$\begin{array}{lll} (x_1 \veebar x_2 \veebar x_3), & (\neg x_1 \veebar x_2 \veebar x_3), & (x_1 \veebar \neg x_2 \veebar x_3), \\ (x_1 \veebar x_2 \veebar \neg x_3), & (x_1 \veebar \neg x_2 \veebar \neg x_3), & (\neg x_1 \veebar x_2 \veebar \neg x_3), \\ (\neg x_1 \veebar \neg x_2 \veebar x_3), & (\neg x_1 \veebar \neg x_2 \veebar \neg x_3) & \end{array}$$

and similarly for larger k. The removal of any one clause in such a problem renders it (classically) satisfiable. This is similar to the problem classes described in (25) and (26), however we maintain problem difficulty in Łukasiewicz logic by restricting truth-value intervals, as further described below.

We observe that, when each clause is required to have truth value exactly 1 but propositions are allowed to have any truth value, `socratic` correctly determines the problem to be

1) unsatisfiable in Gödel logic,
2) satisfiable in Gödel logic when dropping any one clause,
3) trivially satisfiable in Łukasiewicz logic with, e.g., $x_i = .5$,
4) again unsatisfiable in Łukasiewicz logic when propositions are required to have truth values in range either $\left[0, \frac{1}{k}\right)$ or $\left(\frac{k-1}{k}, 1\right]$,
5) and yet again satisfiable in Łukasiewicz logic with constrained propositions when dropping any one clause.

Results are shown in Table 1. We observe that Gödel logic is much slower than Łukasiewicz logic as implemented in `socratic`, likely because it performs mins and maxes between many arguments throughout while Łukasiewicz logic instead performs sums with simpler mins and maxes serving as clamps to the $[0, 1]$ range. Interestingly, the difference between unsatisfiable and satisfiable in Gödel logic is significant; while the satisfiable problems have one fewer clause, this is more likely explained by `socratic` finding a feasible solution quickly. On the other hand, the unsatisfiable and satisfiable problems (with constrained propositions) take roughly the same amount of time for Łukasiewicz logic, though the trivially satisfiable problem is quicker. The exponential increase in runtime with respect to k is mostly explained by the fact that each larger problem has twice as many clauses, but runtime appears to be growing by slightly more than a factor of 2 per each k.

Hájek tautologies. Hájek lists many axioms and tautologies pertaining to a system of logic he describes as basic logic (BL), consistent with a broad class of fuzzy logics, as well as a number of tautologies specific to Łukasiewicz and Gödel logic, all of which should have truth value exactly 1. We implement these tautologies in `socratic` and test whether the empty theory can entail each $(\sigma; \{1\})$ in its respective logic where σ is one of the tautologies. The BL tautologies are divided into batches pertaining to specific operations and properties, specifically axioms, implication, conjunction, min, max, negation, associativity, equivalence, distributivity, and

Table 1. k**-SAT runtimes in seconds for** `socratic` **with different configurations. The columns pertain to items 1 through 5 above.**

k	Gödel unsat.	Gödel satisf.	Łuka. trivial	Łuka. unsat.	Łuka. satisf.
3	.012	.011	.014	.019	.014
4	.022	.020	.022	.031	.033
5	.054	.043	.041	.047	.043
6	.121	.107	.064	.104	.098
7	.204	.255	.173	.167	.206
8	.404	.414	.273	.286	.308
9	.861	.881	.507	.539	.554
10	5.46	1.99	1.03	1.11	1.17
11	18.0	4.34	2.09	2.44	2.21
12	33.3	10.9	4.36	5.06	5.01
13	119	25.8	8.72	12.4	12.3
14	696	71.0	18.4	38.0	35.6

the unary Baaz-Monteiro operator $\triangle$ defined by $f_\triangle(s) = 1$ if $s = 1$ else 0. In addition, there are logic-specific batches of tautologies for Łukasiewicz and Gödel logic. Each of the above BL batches complete successfully for both logics and each of the logic-specific batches complete for their respective logics and, as expected, fail for the other logic. The runtime of individual tests are negligible; the entire test suite of 82 tautologies run on both logics completes in just 2.911 seconds.

Boolean logic. We consider a formula σ defined

$$(\varphi \to \psi) \to ((\neg\varphi \to \psi) \to \psi) \qquad [2]$$

which is valid in classical logic but is not valid in either Łukasiewicz or Gödel logic. Conversely, constraining propositions φ and ψ to have 0–1 truth values via the sentences $(\varphi; \{0,1\})$ and $(\psi; \{0,1\})$ into the theory succeeds in entailing σ in either logic.

Stress test. We consider the experimental configuration given by Formula 2 for a query $(\sigma; S)$ with $S = [.5, 1] \cup \bigcup\left\{(\frac{1}{k+1}, \frac{1}{k}) : 2 \le k \le 10{,}000\right\}$ and for $(\varphi; S')$ and $(\psi; S')$ with $S' = 0 \cup \bigcup\left\{(1 - \frac{1}{k}, 1 - \frac{1}{k+1}) : 2 \le k \le 10{,}000\right\}$. We observe the runtime of `socratic` to be just 11.8 seconds for Gödel logic and 9.38 seconds for Łukasiewicz logic. If we instead use closed intervals throughout, measured runtimes are 17.4 seconds for Gödel and 9.29 seconds for Łukasiewicz.

7. Dealing with weights

In some settings, such as LNN (10), weights are assigned to subformulas, where each real-valued weight determines the influence, or importance, of its respective subformula. For example, in the formula $\sigma_1 \veebar \sigma_2$, the weight w_1 might be assigned to σ_1 and the weight w_2 assigned to σ_2. If $0 < w_1 = 2w_2$, this might indicate that σ_1 is twice as important as σ_2 in evaluating the value of $\sigma_1 \veebar \sigma_2$. Although it might seem natural for weights to be nonnegative and sum to 1, this is not required and LNN does not make this assumption.

As an example of a possible way to incorporate weights, assume that we are using Łukasiewicz real-valued logic, where the value of $\sigma_1 \veebar \sigma_2$ is $\min\{1, s_1 + s_2\}$, when s_1 is the value of σ_1 and s_2 is the value of σ_2. If the weights of σ_1 and σ_2 are w_1 and w_2, respectively, and if both w_1 and w_2 are nonnegative, then we might take the value of $\sigma_1 \veebar \sigma_2$ in the presence of these weights to be $\min\{1, w_1 s_1 + w_2 s_2\}$.

We now show how easy it is to incorporate weights into our approach while still preserving its sound and complete axiomatization. To deal with weights, we define an expanded view of what a formula is, defined recursively. Each atomic proposition is a formula. If σ_1 and σ_2 are formulas, w_1 and w_2 are weights, and α is a binary connective (such as &) then $(\sigma_1 \,\alpha\, \sigma_2, w_1, w_2)$ is a formula. Here w_1 is interpreted as the weight of σ_1 and w_2 as the weight of σ_2 in the formula $\sigma_1 \,\alpha\, \sigma_2$. Also, if σ is a formula, and w is a weight, then $(\neg\sigma, w)$ is a formula, where w is interpreted as the weight of σ. We modify our definition of *subformula* as follows. The subformulas of $(\sigma_1 \,\alpha\, \sigma_2, w_1, w_2)$ are σ_1 and σ_2, and the subformula of $(\neg\sigma, w)$ is σ.

If α is a weighted binary connective, then f_α now has four arguments, rather than two. Thus, $f_\alpha(s_1, s_2, w_1, w_2)$ is the value of the formula $(\sigma_1 \,\alpha\, \sigma_2, w_1, w_2)$ when the value of σ_1 is s_1, the value of σ_2 is s_2, the weight of σ_1 is w_1, and the weight of σ_2 in w_2.

Our axiom and inference rules are just as before, except that we modify the definition of a good tuple for Rule 7. In the sentence $(\sigma_1, \ldots, \sigma_k; S)$, let us say that the tuple $(s_1, \ldots, s_k)$ in S is *good* if (a) for weighted binary connective α, we have $s_m = f_\alpha(s_i, s_j, w_1, w_2)$ when σ_m is $(\sigma_i \,\alpha\, \sigma_j, w_1, w_2)$, and (b) for unweighted connectives it is the same as before.

We can extend Theorem 3.4 (soundness and completeness) and Theorem 4.1 (closure under Boolean combinations) to deal with our sentences $(\sigma_1, \ldots, \sigma_k; S)$ that include weights. The proofs go through just as before, where we use the modified notion of good tuple in Rule 7. Thus, we obtain the following theorems.

Theorem 7.1 *Our axiom system for MD-sentences as adapted for weights is sound and complete.*

Theorem 7.2 *Each finite Boolean combination of sentences* $(\sigma_1, \ldots, \sigma_k; S)$ *that include weights is equivalent to a single such sentence.*

What about the decision procedure that we shall give in Section 6? Its use of a polynomial-time algorithm for linear programming continues to work so long as weights w_i are fixed rational constants and the weighting functions are piecewise linear, such as $w_1 s_1 + w_2 s_2$ (possibly including min or max). As a result, the decision procedure and its implementation stand.

8. Issues in treating the values as probabilities

In this section, where we treat the truth values as probabilities, we are not using a standard real-valued logic but instead the rules of probability. We interpret the truth value of each propositional formula φ as being the probability of φ. Assume that we have n atomic propositions $A_1, \ldots, A_n$. There are then 2^n members of the Venn diagram, each given by a formula $B_1 \cap \cdots \cap B_n$, where B_i is either A_i or $\bar{A}_i$, for $1 \le i \le n$, where $\bar{A}_i$ is the complement of A_i. Instead of conditions (a) and (b) in definition of a good tuple for Rule 7, we have new restrictions (a′) and (b′), which say: (a′) If every member of the Venn diagram appears as a formula σ_i in $(\sigma_1, \ldots, \sigma_k, S)$, then the value assigned to each member of the Venn diagram is nonnegative, and the sum of the values of the members of the Venn diagram is 1, and (b′) if every member of the Venn diagram appears as a formula σ_i in $(\sigma_1, \ldots, \sigma_k, S)$, and if the formula σ_j is logically equivalent to the disjoint union of the members $\tau_1, \ldots, \tau_m$ of the Venn diagram, then the value of σ_j is the sum of the values of $\tau_1, \ldots, \tau_m$. In particular, if σ_i is logically false (such as being the conjunction of two different members of the Venn diagram), then the value of σ_i is 0.

Note that this computation in (b′) gives the correct value no matter what probabilistic dependence or independence holds among the atomic propositions. For convenience, if we wish, we can create new variables such as $\varphi_1|\varphi_2$ (whose value, intuitively, is the value for φ_1 given φ_2), and then add a clause to the conditions of a good tuple that says that if c is the sum of the values of the members of the Venn diagram whose disjoint union is logically equivalent to $\varphi_1 \cap \varphi_2$, if d is the sum of the values of the members of the Venn diagram whose disjoint union is logically equivalent to φ_2, and if $d \neq 0$, then the value of $\varphi_1|\varphi_2$ is c/d. This is useful in Bayesian nets, where the probability of an event is dependent on the probability of its parents.

The new inference rule that is our modification of Rule 7 is clearly sound, and the proof of completeness goes through as before, but using our new notion of a good tuple. Just as we closed under subformulas before applying Rule 7 in the completeness proof earlier, here we include every member of the Venn diagram in the MD-sentence in the proof of completeness.

Also, by a similar argument to that in the proof of Theorem 4.1, we obtain closure under Boolean combinations. We thus have the following two theorems, analogous to Theorems 7.1 and 7.2.

Theorem 8.1 *Our axiom system for MD-sentences as adapted for probabilities is sound and complete.*

Theorem 8.2 *Each finite Boolean combination of sentences* $(\sigma_1, \ldots, \sigma_k; S)$ *that deal with probabilities is equivalent to a single such sentence.*

Note that we are *not* requiring that every sentence contains as formulas every member of the Venn diagram, just as we did not require in the propositional case that every sentence is closed under subformulas. Instead, just as in the completeness argument in the propositional case where we passed in the proof using the axiomatization to a sentence closed under subformulas, here we pass in the proof of completeness using the axiomatization to a sentence that contains all members of the Venn diagram. Thus, the fact that we are making use of the Venn diagram is "behind the curtains" – the user need not know this when writing his sentences. Of course, if the user applies Rule 7 himself, then he needs to be aware of the Venn diagram.

Finally, we note that our sound and complete axiomatization can give us a decision procedure analogous to that in Section 6. In the special case where each atomic proposition A_i is assigned a fixed value a_i, Hailperin (27) gives a decision procedure that is essentially based on the Venn diagram.

9. Related work

Rosser (28) comments on the possibility of considering formulas whose value is guaranteed to be at least θ. For example, if $f_{\veebar}(s_1, s_2) = \max\{s_1, s_2\}$ and $f_{\neg}(s) = 1 - s$, then the truth value of $A \veebar \neg A$ is always at least 0.5. But Rosser rejects this approach, since he notes that there are uncountably many choices for θ, but only countably many recursively enumerable sets (and an axiomatization would give a recursively enumerable set of valid formulas).

Belluci (29) investigates when the set of formulas with values at least θ is recursively enumerable. Font et al. (30) consider the question of what they call "preservation of degrees of truth". They give a method for deciding, for a fixed θ, if σ having a value at least θ implies that φ has value at least θ.

Novák (31) considered a logic with sentences that assign a truth value to each formula of first-order real-valued logic. Thus, using our notation, his sentences would be of the form $(\varphi; \{\theta\})$, where φ is a formula in first-order real-valued logic, and θ is a single truth value. He gave a sound and complete axiomatization.

Another interesting logic is rational Pavelka logic (RPL), an expansion of the standard Łukasiewicz logic where rational truth-constants are allowed in formulas. For example, if r is a rational number, then the formula $r \to \varphi$ says that the value of φ is at least r, and the formula $\varphi \to r$ says that the value of φ is at most r. Therefore, this logic can express the MD-sentences $(\varphi; S)$, when S is the union of a finite number of closed intervals. However, it cannot express strict inequalities. For example, it cannot express that the value of φ is strictly greater than 0.5.[†] This drawback can be solved (20) by expanding the logic with the Baaz-Monteiro $\triangle$ operator (given $\triangle x = 1$ if $x = 1$ and $\triangle x = 0$ otherwise). Such an extension keeps finite-strongly completeness (for Łukasiewicz logic). RPL was introduced by Hájek in (13) as a simplification of the system proposed by Pavelka in (32) in which the syntax contained a truth-constant for each real number of the interval [0,1]. Hájek showed that an analogous logic could be presented as an expansion of Łukasiewicz propositional logic with truth-constants only for the rational numbers in [0,1] and gave a corresponding completeness theorem. Moreover, first-order fuzzy logics with real or rational constants have also been deeply studied starting from Novák's extension of Pavelka's logic to a first-order predicate language in (33) (see e.g. (34)).

Each of (35), (36) and (23) give decision procedures that partially cover the situation we allow in Section 6. The former two support only Łukasiewicz logic. The third, like our decision procedure, works for a variety of logics, though it is explicitly established in (23) that their approach does not support discontinuous operators. Accordingly, unlike our decision procedure, their approach does not work for Gödel logic given its discontinuous $\to$ operator.

In addition, (24) and (37) present decision procedures based on satifiability modulo theories (SMT). The former of these implements mNiBLoS, a versatile means of defining and reasoning in a broad class of fuzzy logics as thoroughly considered in (13). Their approach, however, does not inherently support reasoning in terms of truth value intervals as SoCRAtic does for MD-sentences. (37) presents special cases handling using the Z3 SMT solver for Łukasiewicz and Gödel logic and, in particular, for the finite multi-valued cases of these. This specialized approach demonstrates speedup over (24)'s mNiBLoS

[†] This follows from the stronger fact that if $A_1, \ldots, A_r$ are the atomic propositions, φ is a formula, and G is the set of all value assignments to the atomic propositions that give φ the truth value 1, then since the operators of standard Łukasiewicz logic are continuous (and so the value of φ is a continuous function of the value of the atomic propositions), it follows that $\{(g(A_1), \cdots, g(A_r)) \; : \; g \in G\}$ is a closed subset of $[0, 1]^r$. Note that if $r = 0.5$, then even though the formula $A \to r$ has the value 1 when the value a of A is at most 0.5, the negation $\neg(A \to r)$ does not have the value 1 when $a > 0.5$; instead it has the value $a - 0.5$.

but effectively solves a different problem and so is less directly applicable to our task.

There are various papers in the algebraic framework of residuated lattices and the proof-theoretic framework of hypersequents. For example, see (38). Our approach does not seem to extend to such real-valued logics.

10. Conclusions

We give a sound and finite-strongly complete axiomatization for a rich, novel class of multidimensional sentences about real-valued formulas. By being parameterized, our axiomatization covers a large set, including all of the common real-valued logics in the literature. Our axiomatization allows us to include weights on formulas and extends to probabilities. Having multidimensional sentences is the key to the power of our approach. An interesting open problem is to make use of multidimensional sentences in other contexts.

We provide a decision procedure that covers a subset of these real-valued logics. However, decision procedures going beyond this subset remain future work. Further, the procedure shown should be thought of as a baseline or proof of concept only, not intended to be efficient in practice. Designing efficient inference procedures for real-valued logics is a major area for further development.

Our results give us a way to establish such properties for neuro-symbolic systems that aim or purport to perform logical inference with real values. Because Logical Neural Networks (10) are exactly a weighted real-valued logical system implemented in neural network form, an important immediate upshot of our results for the weighted case is that they provide provably sound and complete logical inference for LNN. Such a result has not previously been established for a neuro-symbolic approach to our knowledge. It is an open question as to whether deep learning models trained "in the wild" (i.e., not deliberately as in LNN (11)) achieve logical behavior. While one of our main motivations was to pave the way forward for AI systems, our results are fundamental, filling a long-standing gap in a very old literature, and can be applied well beyond AI.

ACKNOWLEDGMENTS. We are grateful to Marco Carmosino, who improved the writing in this paper by giving us many helpful comments, and to Guillermo Badia, Ken Clarkson, Didier Dubois, Phokion Kolaitis, Carles Noguera, and Henri Prade for helpful comments. Finally, we are grateful to Lluis Godo, Chris Fermüller, and George Metcalfe for confirming the novelty of our approach and for thoroughly reviewing this manuscript and suggesting many insightful revisions.

1. G Boole, *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities.* (Walton and Maberly) Vol. 2, (1854).
2. LA Zadeh, Fuzzy logic and approximate reasoning. *Synthese* **30**, 407–428 (1975).
3. V Novák, A formal theory of intermediate quantifiers. *Fuzzy Sets Syst.* **159**, 1229–1246 (2008).
4. G Epstein, *Multiple-valued logic design: an introduction.* (CRC Press), (1993).
5. R Fagin, A Lotem, M Naor, Optimal aggregation algorithms for middleware. *J. computer system sciences* **66**, 614–656 (2003).
6. L Serafini, Ad Garcez, Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint* **arXiv:1606.04422** (2016).
7. S Badreddine, Ad Garcez, L Serafini, M Spranger, Logic tensor networks. *Artif. Intell.* **303** (2022).
8. SH Bach, M Broecheler, B Huang, L Getoor, Hinge-loss Markov random fields and probabilistic soft logic. *The J. Mach. Learn. Res.* **18**, 3846–3912 (2017).
9. W Cohen, F Yang, KR Mazaitis, TensorLog: A probabilistic database implemented using deep-learning infrastructure. *J. Artif. Intell. Res.* **67**, 285–325 (2020).
10. R Riegel, et al., Logical neural networks. *arXiv preprint* **arXiv:2006.13155** (2020).
11. S Lu, et al., Training logical neural networks by primal-dual methods for neuro-symbolic reasoning in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021.* (IEEE), pp. 5559–5563 (2021).
12. A Di Nola, A Lettieri, On normal forms in Łukasiewicz logic. *Arch. for Math. Log.* **43**, 795–823 (2004).
13. P Hájek, *Metamathematics of fuzzy logic.* (Springer Science & Business Media) Vol. 4, (1998).
14. P Hájek, D Svedja, A strong completeness theorem for finitely axiomatized fuzzy theories. *Tatra Mountains Math. J.* **12**, 213–219 (1997).
15. A Rose, J Rosser, Fragments of many-valued sentential calculi. *Trans. Am. Math. Soc.* **87**, 74–80 (1958).
16. CC Chang, Algebraic analysis of infinite valued logic. *Trans. Am. Math Soc.* **88**, 467–490 (1958).
17. CC Chang, A new proof of the completeness of the Łukasiewicz axioms. *Trans. Am. Math Soc.* **93**, 74–90 (1959).
18. G Takeuti, S Titani, Fuzzy logic and fuzzy set theory. *Arch. for Math. Log.* **32**, 1–32 (1992).
19. N Rescher, *Many-valued logic.* (McGraw-Hill), (1969).
20. F Esteva, L Godo, E Marchioni, Fuzzy logics with enriched language in *Chapter VIII of the Handbook of Mathematical Fuzzy Logic, Vol 2. Petr Cintula, Petr Hájek, and Carles Noguera (eds.).* (College Publications), pp. 627–711 (2011).
21. NV Murray, E Rosenthal, Signed formulas: A liftable meta-logic for multiple-valued logics in *International Symposium on Methodologies for Intelligent Systems.* (Springer), pp. 275–284 (1993).
22. M Baaz, R Zach, Compact propositional Gödel logics in *Proceedings of 28th International Symposium on Multiple Valued Logic.* (IEEE Computer Society Press), pp. 108–113 (1998).
23. R Hähnle, Many-valued logic and mixed integer programming. *Annals Math. Artif. Intell.* **12**, 231–263 (1994).
24. A Vidal, MNiBLoS: a SMT-based solver for continuous t-norm based logics and some of their modal expansions. *Inf. Sci.* **372**, 709–730 (2016).
25. M Bofill, F Manyà, A Vidal, M Villaret, New complexity results for łukasiewicz logic. *Soft Comput.* **23**, 2187–2197 (2019).
26. S Preto, F Manyà, M Finger, Benchmarking łukasiewicz logic solvers with properties of neural networks in *2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL).* (IEEE), pp. 158–163 (2023).
27. T Hailperin, Best Possible Inequalities for the Probability of a Logical Function of Events. *Am. Math. Mon.* **72**, 343–359 (1965).
28. JB Rosser, Axiomatization of infinite valued logics. *Logique et Analyse* **3**, 137–153 (1960).
29. L Belluce, Further results on infinite valued predicate logic. *J. Symb. Log.* **29**, 69–78 (1964).
30. JM Font, ÀJ Gil, A Torrens, V Verdu, On the infinite-valued Łukasiewicz logic that preserves degrees of truth. *Arch. Math. Log.* **45**, 835–868 (2006).
31. V Novák, Fuzzy logic with extended syntax. *Handb. Math. Fuzzy Log.* **3**, 1063–1104 (2015).
32. J Pavelka, On fuzzy logic i, ii, iii. *Zeitschrift fur Math. Logik und Grundlagen der Math.* **29**, 45—52, 119–134, 447–464 (1979).
33. V Novák, On the syntactico-semantical completeness of first-order fuzzy logic part I (syntax and semantic), part II (main results). *Kybernetika* **26**, 47–66, 134–154 (1990).
34. F Esteva, L Godo, C Noguerra, First-order t-norm based fuzzy logics with truth-constants: distinguished semantics and completeness properties. *Annals Pure Appl. Log.* **161**, 185–202 (2009).
35. G Beavers, Automated theorem proving for Łukasiewicz logics. *Studia Logica* **52**, 183–195 (1993).
36. D Mundici, A constructive proof of McNaughton's theorem in infinite-valued logic. *The J. Symb. Log.* **59**, 596–602 (1994).
37. C Ansótegui, M Bofill, F Manyà, M Villaret, Automated theorem provers for multiple-valued logics with satisfiability modulo theory solvers. *Fuzzy Sets Syst.* **292**, 32–48 (2016).
38. G Metcalfe, F Montagna, Substructural fuzzy logics. *J. Symb. Log.* **72**, 834—864 (2007).