On Modal Logic Formulae Minimization

Giovanni Pagliarini[1], Andrea Paradiso[1], Guido Sciavicco[1] and Ionel Eduard Stan[2]

[1]University of Ferrara, Italy
[2]Free University of Bozen-Bolzano, Italy

Abstract

From the intricate circuits of digital devices to the abstract realms of logical theory, formula minimization remains a cornerstone challenge with various implications. This paper explores the adaptation of formula minimization techniques to modal logic $\mathcal{K}$, driven by the challenges of processing increasingly complex data structures. Our investigation begins by analyzing the inherent computational complexity of minimizing modal formulae, a critical step towards developing effective solutions. We, then, introduce a heuristic algorithm for the purpose. We prove its correctness, addressing the unique demands of modal logic. The implications of this work extend beyond theoretical interest, and include practical insights for the post-hoc treatment of symbolic models in modal logic $\mathcal{K}$, as well as temporal, spatial, and other non-classical logics in advanced computational contexts.

Keywords

Modal logic, Formula minimization, Heuristic algorithm

1. Introduction

The expressiveness of two-valued *Boolean algebra* capable of describing *switching circuit* operations, highlighted by Shannon in 1938 [1], allowed the arising of a new field called *logic synthesis*. The great interest in circuits development, particularly in their Boolean form, highlighted a natural duality between *Boolean circuits* and *Boolean formulae*; Boolean formulae can be considered a particular representation of Boolean circuits. The need for ever smaller circuits and formula representations led to the study of techniques whose goal was to reduce the dimension of a given circuit, preserving its semantic behavior. To solve this problem, different algorithms were theorized, the most famous of which is the Quine–McCluskey algorithm [2], considered a milestone in the field of *logic optimization*.

The challenge of reducing a Boolean circuit to its minimal equivalent can be adeptly transformed into a logical problem using *propositional logic*. Specifically, given a Boolean formula φ and a metric for its size (e.g., the number of tokens in the formula), the goal is to discover a formula φ' that is both equivalent to φ and minimal in terms of size. This problem has been extensively examined when the formulae are represented in *disjunctive normal form* (DNF) or *conjunctive normal form* (CNF), and we refer to it as the *Propositional Minimal Equivalent Formula (PMEF) problem*, addressing it in both its search and decision formats. The decision variant, in particular, is framed as follows: given a propositional formula φ and an integer k, is there an equivalent formula ψ, whose size does not exceed k? Regardless of whether the formula is presented in CNF, DNF, or another generic form, the complexity of PMEF is determined as Σ_2^p-complete, as documented by [3]. This classification places it in the second level of the polynomial hierarchy [4], distinct from the satisfiability problem (*PSAT*), which well-knowingly Σ_1^p-complete, that is, NP-complete.

Diverse strategies have been developed to address the PMEF problem, which can be categorized into *exact* and *heuristic* algorithms. Exact methods inevitably grapple with the problem's inherent exponential time complexity yet consistently deliver globally minimal solutions. In contrast, heuristic techniques strive to bypass the limitation of the required computational effort by offering solutions that are good, though not necessarily optimal and represent local optima. Historically, exact algorithms

CILC 2024: 39th Italian Conference on Computational Logic, June 26-28, 2024, Rome, Italy
giovanni.pagliarini@unife.it (G. Pagliarini); andrea.paradiso@edu.unife.it (A. Paradiso); guido.sciavicco@unife.it (G. Sciavicco); ioneleduardstan@unibz.it (I. E. Stan)
0000-0002-8403-3250 (G. Pagliarini); 0000-0002-3614-2487 (A. Paradiso); 0000-0002-9221-879X (G. Sciavicco); 0000-0001-9260-102X (I. E. Stan)

have evolved from the foundational Quine-McCluskey algorithm. Notable implementations include ESPRESSO [5], which operates in both exact and heuristic capacities, and SCHERZO [6], a state-of-the-art solution. Prominent examples of heuristic algorithms are MINI [7], BOOM [8], and ESPRESSO, again, which can function in both modalities. Moreover, recent advancements include a new algorithm proposed in [9], which minimizes propositional formulae without translating into normal forms (CNF or DNF).

Beyond its foundational application in circuit minimization, propositional minimization finds relevance in several domains. Notably, in the emerging field of *eXplainable AI* (XAI), this technique aids in the post-hoc analysis of models, particularly in symbolic learning, where models are interpreted as sets of propositional formulae. Such minimization processes are crucial, especially as outlined in [10], where the role of prime implicants, a concept linked to formulae minimization, underscores their significance in refining explanations within AI systems.

While minimization has traditionally centered on propositional logic, similar inquiries can be extended to more expressive logical frameworks. For instance, first-order logic has been explored in this context [11]. Our focus here is on *modal logic*, which augments propositional logic's expressive power without venturing into first-order complexities (refer to [12] for an overview). Modal logic is pivotal, particularly as a foundational framework for more complex languages, including *temporal* [13, 14, 15], *spatial* [16], and various *description* logics [17]. By examining the basic unary modal logic $\mathcal{K}$, significant insights can be gained, as evidenced by recent advances in symbolic learning beyond propositional logic [18, 19, 20, 21], which are already apparent at this fundamental level [22]. Similarly to the propositional case, machine learned modal logic models—conceptualized as sets of modal formulae—can benefit significantly from efficient minimization algorithms, especially since modal logic lacks a standardized normal form, differentiating it markedly from propositional and first-order logics.

In this work, we introduce the *Modal Minimal Equivalent Formula (MMEF) problem*, defined as the search for the smallest modal formula equivalent to a given one. Its decision version asks whether a modal formula ψ exists, equivalent to φ and not exceeding a specified size k. We establish that MMEF is PSPACE-complete, aligning its complexity with modal logic $\mathcal{K}$ satisfiability testing (*MSAT*). This classification leads to a naïve but optimal exact algorithm that systematically explores all potential smaller formulae within the original signature and verifies each for equivalence. The primary goal of this paper is to propose an innovative heuristic algorithm for MMEF that leverages insights from its propositional counterparts and to validate its correctness. As we will demonstrate, this heuristic is uniquely adaptable and suitable for any extension of basic modal logic, regardless of the complexity or variety of modal operators employed.

This work is organized as follows. In Section 2 we briefly recall the basic modal logic, its syntax, and its semantics. In Section 3 we define and study MMEF. Then, in Section 4 we describe our heuristic solution, before concluding.

2. Modal Logic

The syntax of well-formed formulae in the basic modal language $\mathcal{K}$ [12], denoted later by $\varphi, \psi, \ldots$ (the Greek letters $\alpha, \beta, \ldots$, instead, will be used to denote numeric parameters), is based on a set of *propositional letters* $\mathcal{P} = \{p, q, \ldots\}$. Formulae are built with the following grammar:

$$\varphi \quad ::= \quad \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi,$$

where the missing classic Boolean operators can be obtained as shortcuts.

The symbols $\Diamond$ and $\Box$ are called *modalities*, and, informally, they are treated as existential and universal operator, respectively. A formula in which no modality occurs is said to be *purely propositional*, or *Boolean*. Given two formulae φ, ψ, we use $\varphi = \psi$ to denote the fact that they are *syntactically identical*, and $|\varphi|$ to denote the *size* of φ. Several different, but substantially equivalent, size measures of formula can be defined; the most common one is the number of tokens, that is, its *length*. In the

following, we shall assume that $|\varphi|$ denotes precisely the length, but all our results immediately apply to the other possibilites. The subset of $\mathcal{P}$ that is used in a formula φ, plus the symbols $\top$ and $\bot$, is called *signature* of φ, and denoted by $sig(\varphi)$; given a signature $\mathcal{S} \subseteq \mathcal{P} \cup \{\top, \bot\}$, we denote by $\Phi(\mathcal{S})$ the set of all well-formed modal formula that can be built on it. Finally, we denote by $sub(\varphi)$ the set of all sub-formulae of a formula φ; a formula $\psi \in sub(\varphi)$ such that its outermost connective is a modal operator, that is, a *modal sub-formula*, is said to be a *maximally modal sub-formula* if every sub-formula $\xi \in sub(\varphi)$ such that $\psi \in sub(\xi)$ has, as outermost connective, a Boolean one, and the subset of $sub(\varphi)$ of all and only maximally modal sub-formulae of a given formula φ is denoted by *msub*(φ).

In the so-called *Kripke* semantics, we interpret a formula on a *Kripke interpretation* $K = (W, R, V)$, where the pair (W, R), in which W is a non-empty, finite set of *worlds* and $R \subseteq W \times W$ is the *accessibility relation*, is called *Kripke frame*, and V is a *valuation* function that assigns to each world the set of propositional letters that are true on it, that is $V : W \to 2^{\mathcal{P}}$. In general, we denote Kripke interpretations with the symbols $K, K_1, \ldots$. Given an interpretation K and a world w, we say that K, w *satisfies* a formula φ, denoted by $K, w \models \varphi$, if and only if

$$
\begin{array}{lll}
K, w \models p & \text{iff} & p \in V(w), \\
K, w \models \neg\varphi & \text{iff} & K, w \not\models \varphi, \\
K, w \models \varphi \wedge \psi & \text{iff} & K, w \models \varphi \text{ and } K, w \models \psi, \\
K, w \models \varphi \vee \psi & \text{iff} & K, w \models \varphi \text{ or } K, w \models \psi, \\
K, w \models \Diamond\varphi & \text{iff} & \exists v \text{ s.t. } wRv \text{ and } K, v \models \varphi, \\
K, w \models \Box\varphi & \text{iff} & \forall v \text{ s.t. } wRv \text{ it is the case that } K, v \models \varphi.
\end{array}
$$

Moreover, we have

$$
\begin{array}{l}
K, w \models \top, \\
K, w \not\models \bot.
\end{array}
$$

Given a modal formula φ, we say that it is *satisfiable* if and only if there exists a Kripke interpretation K and a world w such that $K, w \models \varphi$; in this case, K is said to be a *model* of φ (however, we often use the term *model* to refer to machine learning models). In the following, we use the notation $\varphi \equiv \psi$ to denote the fact that φ and ψ are *equivalent*, that is, for every interpretation K and world w, it is the case that $K, w \models \varphi$ if and only if $K, w \models \psi$. Moreover, it is well-known that every modal logic formula can be presented in *negated normal form* (*NNF*), that is, in such a way that the symbol $\neg$ only occurs in front of propositional letters, and that converting a formula into an equivalent one in NNF does not increase its length by more than a constant factor; in the following, we assume that every formula is in NNF, and we use NNF() to denote an algorithm that turns a given formula in its negated normal form equivalent. A propositional letters or the negation of a propositional letter is known as a *literal*; here, we denote literals by $\lambda, \lambda_1, \ldots$. Finally, given two formulae φ, ξ, and given a formula $\psi \in sub(\varphi)$, by $\varphi[\psi/\xi]$ we denote the formula φ after having syntactically replaced every occurrence of the sub-formula ψ in φ with ξ. Observe that modal logic generalizes propositional logic, and the valuation function associates a propositional interpretation to each world. Later in this paper, we shall use $M, M_1, \ldots$ to denote propositional interpretations.

The *satisfiability* problem for modal logic (*MSAT*), that is, the problem of establishing if, given a modal formula, there exists a Kripke interpretation and a world that satisfy it, is known to be PSPACE-complete [23]; this is in sharp contrast with the same problem at the propositional level (*PSAT*), which is NP-complete. There exist tens of available systems for satisfiability checking of modal formulae. These are generally separated into *tableau-based* systems (e.g., [24, 25]), which can be considered extensions of classical propositional tableaux, *DPLL-based* methods, that is, methods based on the progressive substitution of modal sub-formulae with fresh propositional letters, so that standard propositional satisfiability checkers can be then used (e.g., [26]), *translational* approaches, based on the idea of translating a modal formula into its first-order counterpart, so that standard first-order satisfiability checkers can be then used (e.g., [27]), and *automata-based* techniques (e.g., [28]); an excellent survey on this topic can be found in [29]. In the following, we shall use *PSAT()* (resp., *MSAT()*) to denote an algorithm that solves an instance of the satisfiability problem for propositional (resp., modal) logic.

Modal logic is paradigmatic for several varieties of more-than-propositional logics. Indeed, most classic temporal [13, 14, 15], spatial [16], and description [17] logics are specializations of modal logics with more than one accessibility relations (possibly with higher arities) and associated modalities, subject to constraints on the admitted Kripke interpretations, that range from very simple and intuitive ones (e.g., transitivity, antisymmetry) to very complex ones (e.g., when worlds are assumed to be intervals and modalities are assumed to mimic relations between intervals).

3. Exact Minimization of Modal Formulae

At the propositional level, the problem of minimizing a given formula has been presented in several forms. In its decision version it is usually defined as the problem, given a formula φ and an integer k, of establishing whether a formula ψ, smaller than or of the same size of φ, and equivalent to φ, exists. Typically, propositional formulae are assumed to be in *disjunctive normal form* (*DNF*) or *conjunctive normal form* (*CNF*), but the problem can be defined for generic formulae as well; as a matter of fact, this problem is studied in [9]. Regardless of the form of the input formula, this problem, called here the *Propositional Minimal Equivalent Formula (PMEF) problem*, is known to be Σ_2^p-complete [3]. In the following, we shall use *PMEF()* to denote both an algorithm that solves an instance (φ, k) of the decision version of this problem and an algorithm that solves an instance φ of its search counterpart. Whenever convenient, we shall use *EXACT-PMEF()* (resp., *HEURISTIC-PMEF()*) to denote an exact (resp., heuristic) algorithm for solving a particular instance.

Generalizing the minimization problem to modal formulae is very natural. However, modal formulae do not have an accepted disjunctive/conjunctive normal form in the same way that propositional ones do; normal forms for modal formulae have been studied in [30], but the available proposals lack the naturalness and simplicity that are characteristic of normal forms.

Definition 1. *Given a modal formula φ and an integer k, the* Modal Minimal Equivalent Formula (MMEF) problem *consists in deciding whether there exists a formula ψ such that $\psi \equiv \varphi$ and its size is at most k.*

In the following, we shall use the same symbol to denote both the decision and the search version of this problem. Similarly to the propositional case, in its search version this problem too can be solved exactly and heuristically, and we shall use *EXACT-MMEF()* (resp., *HEURISTIC-MMEF()*) to denote a generic exact (resp., heuristic) algorithm for it.

Lemma 1. *MMEF is PSPACE-hard.*

Proof: We aim to establish the PSPACE-hardness of MMEF through a Turing reduction from MSAT. Consider an instance φ of MSAT. Notably, φ is unsatisfiable if and only if it is equivalent to the formula $\bot$. Let us define $\psi = \varphi \wedge p \wedge q$, where p and q are propositional letters not included in $sig(\varphi)$, the signature of φ; we convert the instance φ for MSAT into the instance $(\psi, 1)$ for MMEF.

Firstly, assume φ is satisfiable. Under this condition, any formula ξ equivalent to φ and minimal in size cannot be shorter than a single syntactic token (i.e., ξ must be equivalent to $\top$ or to a single propositional letter), and, in particular, cannot be p nor q. Consequently, any minimal size formula ζ equivalent to ψ must, at least, include the conjunction $p \wedge q$. Therefore, querying MMEF(ψ, 1) under these conditions would return *false*. Conversely, if φ is unsatisfiable, it is equivalent to $\bot$, and so is ψ. In this scenario, MMEF(ψ, 1) would return *true*, indicating that a minimal formula equivalent to ψ and smaller than or equal to size 1 exists.

This construction shows that MSAT $\leq_T$ MMEF. Given that MSAT is known to be PSPACE-hard, it follows that MMEF is also PSPACE-hard. □

As a side observation, it is easy to see that the above argument could be generalized to prove that the minimization problem in any non-trivial logic is at least as hard as its satisfiability problem.

Algorithm 1: Exact Modal Minimal Equivalent Formula

```
1  function EXACT-MMEF(φ):
2      forall k ≤ |φ| do
3          forall ψ ∈ Φ(sig(φ)) of length k do
4              if EQUIVALENT(φ, ψ) then
5                  return ψ
6              end
7          end
8      end
9      return φ
10 end
11 function EQUIVALENT(φ, ψ):
12     return not MSAT((φ ∧ ¬ψ) ∨ (ψ ∧ ¬φ))
13 end
```

As it turns out, MMEF is also in PSPACE. This can be proved by providing a sound and complete PSPACE algorithm for the search version of MMEF, that is, EXACT-MMEF(), shown in Alg. 1. Observe, in particular, line 2 and 3 of EXACT-MMEF(): our systematic, ordered exploration of the search space for $\psi \in \Phi(sig(\varphi))$ guarantees that this step does not require more than polynomial space, as it can be implemented without memorizing smaller formulae.

Lemma 2. *EXACT-MMEF() is terminating, sound and complete. Specifically, if* $\psi =$ *EXACT-MMEF*(φ)*, then:*

(i) $\psi \equiv \varphi$ *and*
(ii) for every formula modal formula ξ*, if* $|\xi| < |\psi|$*, then* $\xi \not\equiv \varphi$*.*

Proof: It is immediate to observe that, for a given φ, the execution EXACT-MMEF(φ) always terminates.

Suppose $\psi =$ EXACT-MMEF(φ). This implies that the query MSAT($(\psi \wedge \neg\varphi) \vee (\varphi \wedge \neg\psi)$) returns *false*, indicating that $(\psi \wedge \neg\varphi) \vee (\varphi \wedge \neg\psi)$ is unsatisfiable. Consequently, the equivalence $\psi \leftrightarrow \varphi$ holds, verifying that $\psi \equiv \varphi$ and establishing (i).

For (ii), assume by contradiction that there exists a modal formula ξ such that $|\xi| < |\psi|$ and $\xi \equiv \varphi$. Given that $|\xi| < |\psi|$, ξ would have been tested for equivalence with φ before ψ. But since $\xi \equiv \varphi$, then MSAT($(\xi \wedge \neg\varphi) \vee (\varphi \wedge \neg\xi)$) would have returned *false*, so EXACT-MMEF(φ) would have returned ξ, which is in contradiction with our assumption that EXACT-MMEF(φ) has returned ψ. Therefore no such ξ can exist, concluding the proof. □

Lemma 3. *MMEF is in PSPACE.*

Proof: Consider the operation of EXACT-MMEF(). The algorithm functions within polynomial space bounds, as it tests each formula sequentially and discards each partial result; the space requirement for each test is also polynomial. To solve a decision instance (φ, k) for MMEF, EXACT-MMEF() is executed on φ. Subsequently, the size of the resulting formula is compared to the integer k. Since EXACT-MMEF() operates within polynomial space and the size comparison also requires only a constant amount of additional space, MMEF is PSPACE. □

Theorem 1. *MMEF is PSPACE-complete.*

It is worth observing that our approach for EXACT-MMEF() is substantially equivalent to the naïve approach for EXACT-PMEF() as shown in [9], used by the authors as baseline.

Algorithm 2: Heuristic Modal Minimal Equivalent Formula

```
function HEURISTIC-MMEF(φ, α, β, γ):
    return HMMEF(φ, α, β, γ, |φ|, sig(φ))
end
function HMMEF(φ, α, β, γ, Max, S):
    if |φ| ≤ α then
        return EXACT-MMEF(φ)
    else
        if φ = φ1 ⊙ φ2 then
            (ψ, H) ← PROPOREPLACE(φ, γ, ∅, Max, S)
            if |ψ| ≤ β then
                ψ' ← EXACT-PMEF(ψ, γ)
            else
                ψ' ← HEURISTIC-PMEF(ψ, γ)
            end
            (φ, H) ← PROPOREPLACE(NNF(ψ'), γ, H^-1, 0, S)
        end
        if φ = φ1 ⊙ φ2 then
            return HMMEF(φ1, α, β, γ, Max, S) ⊙ HMMEF(φ2, α, β, γ, Max, S)
        else if φ = ⊠φ1 then
            return ⊠HMMEF(φ1, α, β, γ, Max, S)
        end
    end
end
```

4. Heuristic Minimization of Modal Formulae

The naïve PSPACE approach to formula minimization, as discussed, exhibits considerable inefficiency. This inefficiency stems primarily from three sources: unavoidable redundant exploration of formulae equivalent to those already tested, inherently exponential time required for each test, and exponentially wide search space. While the first issue could be partially mitigated by incorporating sorting techniques to organize formulae, and the second one by employing very efficient implementations of MSAT solvers, the number of potential candidates remains the most relevant bottleneck of this approach.

To develop a more practical solution, we consider a heuristic approach that does not ensure a global optimum but offers more operational feasibility. Drawing inspiration from [26], this approach exploits efficient solutions to the PMEF problem. By progressively applying consistent substitutions, we aim to minimize the initial formula at least from a propositional standpoint. It is important to note that propositional minimization generally surpasses mere satisfiability checking in complexity, and most current methods require the formula to be in a specific normal form. However, as demonstrated in [9], it is feasible to minimize propositional formulae without converting them to normal form. Alternatively, one could temporarily transform propositional sub-formulae into CNF or DNF for exact or heuristic minimization, and subsequently assessing whether the resultant formula is indeed smaller than the original. Throughout this process, we transition from exact to heuristic minimization methods based on practical considerations. To ensure consistency in substitutions, an external hash table keyed by the formulae themselves is utilized. This structure also aids in identifying semantically opposite sub-formulae, a task that can be executed through exact or heuristic methods. For modal sub-formulae that are sufficiently short, exact minimization remains feasible. In summary, our recursive, heuristic approach hinges on three key parameters: α, the maximum length of a modal sub-formula below which exact minimization is employed (as opposed to continuing recursion in search of smaller alternatives); β, the cutoff length for employing exact over heuristic minimization for propositional sub-formulae; and γ, the threshold under which exact substitutions are preferred to heuristic substitutions.

HEURISTIC-MMEF(), outlined in Algorithm 2, serves as a simple driver for the recursive HMMEF() function. The parameters α, β, and γ are natural numbers that modulate the optimizations within the algorithm. Initially, HMMEF() checks if the input formula φ is small enough ($|\varphi| \leq \alpha$); if so, EXACT-

MMEF() is invoked directly. If $|\varphi|$ exceeds α, HHMEF() evaluates the structure of φ. For formulae where the outermost connective is modal (i.e., $\varphi = \boxtimes\psi$ with $\boxtimes \in \{\Diamond, \Box\}$), HHMEF() recursively calls itself on its argument (i.e., ψ). If, on the other hand, the outermost connective is Boolean (i.e., $\varphi = \psi_1 \circledcirc \psi_2$), the process resorts to Boolean minimization.

PROPOREPLACE() utilizes a hash table H, keyed by modal formulae, to manage substitutions. For a given formula φ and any sub-formula $\psi \in sub(\varphi)$, $H[\psi]$ stores the literal λ replacing ψ in φ after substitution. The goal of PROPOREPLACE() is to recursively substitute sub-formulae in φ with fresh literals or their negations to render the formula purely propositional. Each execution begins with an empty H. In the simplest case where φ is already a literal, no substitution is necessary. If the outermost operator is modal and φ is sufficiently short ($|\varphi| \leq \gamma$), PROPOREPLACE() explores all smaller potential equivalent formulae within φ's signature. If a formula ψ is found to be semantically equivalent to φ (or $\neg\varphi$), PROPOREPLACE() checks if a literal λ has already been assigned to it and returns that literal (or its negation). If no such ψ is found (including the case where ψ is φ itself), indicating that φ has not been substituted previously, a fresh literal is assigned, updating H with both φ and $\neg\varphi$. This substitution is deemed exact. Conversely, if φ is too lengthy, PROPOREPLACE() repeats these steps but limits its scope to checking if H already contains φ or its negated form in NNF. If absent, a new fresh literal is assigned, noting that this substitution may not be optimal as it might replace two equivalent sub-formulae with different literals. When the outermost operator of φ is Boolean, PROPOREPLACE() recursively calls itself on the inner sub-formulae, performs the replacements, and reconstructs the formula. Here, H may remain unchanged but could be updated during the recursive calls.

Because HEURISTIC-MMEF() is not an exact algorithm, we can only prove its soundness; to this end, we first need to prove the soundness of PROPOREPLACE(). In the following, we assume that HEURISTIC-MMEF() is called on some formula φ with $1 \leq \alpha, \beta, \gamma \leq |\varphi|$, that *Max* is the length of the formula onto which HEURISTIC-MMEF() is called, and that $\mathcal{S}$ its signature.

Lemma 4. *If* $(\psi, H) = PROPOREPLACE(\varphi, \gamma, \emptyset, Max, \mathcal{S})$ *and* $msub(\varphi) = \{\psi_1, \ldots, \psi_n\}$*, then:*

(i) ψ *is Boolean,*
(ii) $\psi = \varphi[\psi_1/H[\psi_1], \ldots, \psi_n/H[\psi_n]]$*, and*
(iii) if $H[\psi_i] = H[\psi_j]$ *(resp.,* $H[\psi_i] = \neg H[\psi_j]$*) for some* $i \neq j$*, then* $\psi_i \equiv \psi_j$ *(resp.,* $\psi_i \equiv \neg\psi_j$*).*

Proof: We proceed by double induction on the structural complexity of φ and the recursive depth of the call to prove a stronger property. If $(\psi, H') = \text{PROPOREPLACE}(\varphi, \gamma, H, Max, \mathcal{S})$, then:

(i) $\psi = \varphi[\xi_1/H'[\xi_1], \ldots, \xi_n/H'[\xi_n]]$ for all maximal modal sub-formulae $\xi_1, \ldots, \xi_n$ of φ, unless φ is a literal,
(ii) if ψ is a literal not in $\mathcal{S}$, then $H'[\psi]$ exists,
(iii) for any pair of formulae ξ', ξ'', $H'[\xi'] = H'[\xi'']$ (resp., $H'[\xi'] \equiv \neg H'[\xi'']$) implies $\xi' \equiv \xi''$ (resp., $\xi' \equiv \neg\xi''$), and
(iv) for any formula ξ, $H[\xi]$ maps to a literal that is either $\tilde{t}$ or $\neg\tilde{t}$, where $\tilde{t} \notin \mathcal{S}$.

Assume that φ is a literal, that is, $\varphi = p$ or $\varphi = \neg p$; by construction, it must be the case that $p \in \mathcal{S}$. Then, PROPOREPLACE() returns $(\psi, H') = (\varphi, H)$. If this is the first call to PROPOREPLACE(), then both H and H' are empty, and if it is not, then H remains unchanged from the previous call; in both cases, (i) – (iv) trivially hold. This solves the base case along both dimensions, that is, the call number and the complexity of the input formula; all other cases are inductive.

If the outermost connective of φ is a modal operator, we examine cases based on the size of φ relative to γ. For φ with size less than or equal to γ, all smaller NNF formulae within the signature $sig(\varphi)$ are considered. If a formula ξ is such that $\xi \equiv \varphi$ (resp., $\xi \equiv \neg\varphi$) exists with preassigned $H[\xi]$, we return $(H[\xi], H)$ (resp., $(H[\neg\xi], H)$). Then, (i) and (ii) are satisfied trivially, and (iii) and (iv) hold by inductive hypothesis. If no such ξ is found, $\tilde{t}$ is introduced, and we return $(\tilde{t}, H')$ setting $H'[\varphi/\tilde{t}, NNF(\neg\varphi)/\neg\tilde{t}]$. Once more, (i) and (ii) hold trivially; (iii) and (iv), instead, come by

Algorithm 3: Propositional Replace

```
function PROPOREPLACE(φ, γ, H, Max, S):
    if φ = p or φ = ¬p then
        if p ∈ S then
            return (φ, H)
        else
            return (H[φ], H)
        end
    else if φ = ⊠φ1 then
        if |φ| ≤ γ then
            forall k ≤ Max do
                forall ψ ∈ Φ(sig(φ)) such that ψ is in NNF and |ψ| = k do
                    if EQUIVALENT(ψ, φ) and H[ψ] exists then
                        return (H[ψ], H)
                    end
                end
            end
            t̃ ← NEWLETTER()
            H[φ] ← t̃
            H[NNF(¬φ)] ← ¬t̃
            return (H[φ], H)
        else
            if H[φ] exists then
                return (H[φ], H)
            else
                t̃ ← NEWLETTER()
                H[φ] ← t̃
                H[NNF(¬φ)] ← ¬t̃
                return (H[φ], H)
            end
        end
    else if φ = φ1 ⊙ φ2 then
        (φ'1, H1) ← PROPOREPLACE(φ1, γ, H, Max, S)
        (φ'2, H2) ← PROPOREPLACE(φ2, γ, H1, Max, S)
        return (φ'1 ⊙ φ'2, H2)
    end
end
```

construction, because $\tilde{t}$ is fresh. If $|\varphi| > \gamma$, then we proceed exactly in the same way, but limiting our search to $\xi = \varphi$ and $\xi = NNF(\neg\varphi)$, leading to exactly the same conclusions.

Lastly, if $\varphi = \varphi_1 \circledcirc \varphi_2$, where $\circledcirc \in \{\wedge, \vee\}$, recursive calls process sub-formulae φ_1 and φ_2. After the call on φ_1 and H, PROPOREPLACE() returns the pair (φ'_1, H_1). After the second call, on φ_2 and H_1, PROPOREPLACE() returns the pair (φ'_2, H_2). Then, the pair $(\psi, H') = (\varphi'_1 \circledcirc \varphi'_2, H_2)$ is returned. At this point, (i) holds because $\mathit{msub}(\varphi) = \mathit{msub}(\varphi'_1) \cup \mathit{msub}(\varphi'_2)$, and (ii) holds trivially. Finally, (iii) and (iv) hold by construction, and by the inductive hypothesis for H_1 and H_2. □

Lemma 5. *If it is the case that* $(\psi, H) = \mathit{PROPOREPLACE}(\varphi, \gamma, \emptyset, \mathit{Max}, \mathcal{S})$, $\psi' = \mathit{PMEF}(\psi)$, *and* $(\psi'', H) = \mathit{PROPOREPLACE}(\psi', \gamma, H^{-1}, \mathit{Max}, \mathcal{S})$, *then* $\psi'' \equiv \varphi$.

Proof: We aim to demonstrate that for any Kripke interpretation $K = (W, R, V)$ and world w, if $K, w \models \varphi$, then $K, w \models \psi''$, and the other way around. Consider a Kripke interpretation K and a world w such that $K, w \models \varphi$. Define $K' = (W, R, V')$ where V' behaves as V on every world and every propositional letter $p \in sig(\varphi)$, while, for a generic world w and letter $\tilde{t} \notin sig(\varphi)$ is defined as follows:

$$\tilde{t} \in V'(w) \text{ iff } \begin{cases} H^{-1}[\tilde{t}] = \xi & \text{and} \quad K, w \models \xi, \quad \text{or} \\ H^{-1}[\tilde{t}] = \neg\xi & \text{and} \quad K, w \not\models \xi \end{cases}$$

with $\xi \in \mathit{msub}(\varphi)$. Observe, first, that K' is well-defined, thanks to Lemma 4, (iii). Moreover, since $K, w \models \varphi$, by Lemma 4, (ii) and (iii), we know that $K', w \models \psi$. Then, again by Lemma 4, (i), we know that ψ is a propositional formula, so that PMEF() can be applied to it; clearly, the resulting formula ψ' is equivalent to ψ, so that $K', w \models \psi'$. Now, we want to prove that $K', w \models \psi''$, and we proceed by structural induction on ψ'.

If $\psi' = p$ or $\psi' = \neg p$ for $p \in \mathcal{S}$, then $\psi'' = \psi'$ and the claim follows directly. If $\psi' = \tilde{t}$ (or $\neg\tilde{t}$), where $\tilde{t} \notin \mathcal{S}$, then $\psi'' = \xi$ for some $\xi \in \mathit{msub}(\varphi)$ such that $H[\xi] = \tilde{t}$ (or $H[\xi] = \neg\tilde{t}$). Given that $K', w \models \tilde{t}$ (or $K', w \not\models \tilde{t}$), by definition $K', w \models \xi$ follows.

Assume, now, $\psi' = \zeta'_1 \wedge \zeta'_2$. If $K', w \models \psi'$, then both $K', w \models \zeta'_1$ and $K', w \models \zeta'_2$. Since $\psi'' = \zeta''_1 \wedge \zeta''_2$ where ζ''_1, ζ''_2 are the results from a call to PROPOREPLACE() on ζ'_1, ζ'_2, by the induction hypothesis $K', w \models \zeta''_1$ and $K', w \models \zeta''_2$, hence $K', w \models \psi''$. The disjunction case $\psi' = \zeta'_1 \vee \zeta'_2$ is analogous.

Finally, because $K', w \models \psi''$ and $sig(\psi'') = sig(\varphi)$, it follows that $K, w \models \psi''$ as well, thereby concluding that $K, w \models \varphi$. The converse, that $K, w \models \psi''$ implies $K, w \models \varphi$, follows by a similar argument. □

Lemma 6. *If $\psi = \mathit{HHMEF}(\varphi, \alpha, \beta, \gamma, \mathit{Max}, \mathcal{S})$, then $\psi \equiv \varphi$ and $|\psi| \leq |\varphi|$.*

Proof: We use induction on the complexity of φ to establish that if ψ results from any invocation of HHMEF$(\varphi, \alpha, \beta, \gamma, \mathit{Max}, \mathcal{S})$, then $\psi \equiv \varphi$. Additionally, we aim to show that $|\psi| \leq |\varphi|$.

Consider the case when $|\varphi| \leq \alpha$. In this scenario, ψ is derived from the exact minimization of φ. Exact minimization ensures that ψ is equivalent to φ and its size is less than or equal to that of φ, directly fulfilling the lemma's conditions.

Now, let us examine the situation where $|\varphi| > \alpha$. If φ takes the form $\varphi_1 \circledcirc \varphi_2$, then after one substitution with PROPOREPLACE(), one propositional minimization, and one back substitution, again, with PROPOREPLACE(), we obtain ψ'. Then Lemma 5 applies, and $\psi' \equiv \varphi$; thanks to the minimization step, $|\psi'| \leq |\varphi|$. At this point, three possibilities arise. If ψ' is a literal, then $\psi = \psi'$, and the lemma is satisfied. If $\psi' = \psi'_1 \circledcirc \psi'_2$, then, in general, ψ' may be different from φ; yet, HMMEF() is called recursively on ψ'_1 and ψ'_2, returning ψ_1 and ψ_2, respectively. By inductive hypothesis, $\psi_1 \equiv \psi'_1$ (resp., $\psi_2 \equiv \psi'_2$) and $|\psi_1| \leq |\psi'_1$ (resp., $|\psi_2| \leq |\psi'_2)$. But then, as we wanted, $\psi \equiv \varphi$ and $|\psi| \leq |\varphi|$. Finally, if $\psi' = \boxtimes\psi'_1$, then again, in general, ψ' may be different from φ; yet, HMMEF() is called recursively on ψ'_1, returning ψ_1 and, by inductive hypothesis, $\psi_1 \equiv \psi'_1$ and $|\psi_1| \leq |\psi'_1$. Therefore, $\psi \equiv \varphi$ and $|\psi| \leq |\varphi|$. □

Theorem 2. *HEURISTIC-MMEF() is sound, that is, if $\psi = \mathit{HEURISTIC\text{-}MMEF}(\varphi, \alpha, \beta, \gamma)$, then $\psi \equiv \varphi$ and $|\psi| \leq |\varphi|$.*

It is quite trivial to see that HEURISTIC-MMEF() works in polynomial space; however, it is worth discussing its temporal complexity. Let $\mathcal{C}_{\mathit{HEURISTIC\text{-}MMEF}}(n, \alpha, \beta, \gamma)$ denote the cost of executing HEURISTIC-MMEF() on a formula of length n, with parameters α, β, and γ, can be now estimated. A straightforward computation of its growth rate might not yield insightful results due to its heavy dependence on several factors: the cost $\mathcal{C}_{\mathit{EXACT\text{-}MMEF}}$ from each of the multiple runs of EXACT-MMEF(), $\mathcal{C}_{\mathit{EXACT\text{-}PMEF}}$ from each of the various executions of EXACT-PMEF(), $\mathcal{C}_{\mathit{HEURISTIC\text{-}PMEF}}$ from the calls to HEURISTIC-PMEF(), and $\mathcal{C}_{\mathit{PSAT}}$ from the applications of PSAT(). Nevertheless, it is possible to see that $\mathcal{C}_{\mathit{HEURISTIC\text{-}MMEF}}(n, \alpha, \beta, \gamma) = \mathcal{O}(\mathcal{C}_{\mathit{EXACT\text{-}MMEF}}(n))$, that is, that the heuristic approach is expected not to perform worse than the exact method.

We establish this by setting α at some given fixed value, while β and γ are configured to their worst-case scenarios. Intuitively, as α increases, HEURISTIC-MMEF()'s behavior more closely approximates that of EXACT-MMEF(). When β and γ are maximized, EXACT-PMEF() is utilized more frequently as opposed to HEURISTIC-PMEF(), which inherently increases computational costs. Additionally, this configuration leads to a higher frequency of PSAT() calls, further influencing the overall computational expense.

Theorem 3. *HEURISTIC-MMEF() is at least as efficient as EXACT-MMEF(), that is, $\mathcal{C}_{\text{HEURISTIC-MMEF}}(n, \alpha, \beta, \gamma) = \mathcal{O}(\mathcal{C}_{\text{EXACT-MMEF}}(n))$.*

Proof: We start by noting that the worst case for HEURISTIC-MMEF() occurs when we set $\beta = \gamma = n$. Here, we analyze its computational complexity under these parameters, denoted as $\mathcal{C}_{\textit{HEURISTIC-MMEF}}()$, and show that it is bounded by $\mathcal{O}(\mathcal{C}_{\textit{EXACT-MMEF}}(n))$.

Consider a modal formula of length n. In the most challenging scenario for HEURISTIC-MMEF(), EXACT-MMEF() is invoked frequently on sub-formulae of length α. This scenario typically arises when n symbols are distributed across the maximum number of sub-formulae, each comprising α symbols; at most, such sub-formulae can be $\frac{n}{\alpha}$. Each application of EXACT-MMEF() on these α-long sub-formulae entails a cost of $\mathcal{O}(\mathcal{C}_{\textit{EXACT-MMEF}}(\alpha))$. Moreover, again in the worst case, we can bound the cost due to the remaining operations as follows. Each sub-formula goes through three steps: a call to PROPOREPLACE() to perform a substitution, a call to EXACT-PMEF() (since $\beta = n$), and a call to PROPOREPLACE() to perform the back substitution. While the last operation only adds $\mathcal{O}(n)$ to the total cost, the first one consists of the entire exploration of the sub-formula ($\mathcal{O}(n)$) plus an examination of each of $\mathcal{O}(2^n)$ potentially equivalent formulae (since $\gamma = n$) and a call to PSAT() ($\mathcal{C}_{\textit{PSAT}}(n)$). Summing up the costs, we derive:

$$\mathcal{C}_{\textit{HEURISTIC-MMEF}}(n, \alpha, \beta, \gamma) = O(\frac{n}{\alpha}\mathcal{C}_{\textit{EXACT-MMEF}}(\alpha) + n(n2^n\mathcal{C}_{\textit{PSAT}}(n) + \mathcal{C}_{\textit{PMEF}}(n) + n)).$$

Since α is a constant, and since each call to EXACT-MMEF(φ), where $|\varphi| = n$, requires $\mathcal{O}(2^n)$ calls to MSAT(), that is, it entails a cost of $\mathcal{O}(2^n\mathcal{C}_{\textit{MSAT}}(n))$, we have that:

$$\mathcal{C}_{\textit{HEURISTIC-MMEF}}(n, \alpha, \beta, \gamma) = O\left(n(n2^n\mathcal{C}_{\textit{PSAT}}(n) + \mathcal{C}_{\textit{PMEF}}(n) + n)\right) = \mathcal{O}(2^n\mathcal{C}_{\textit{MSAT}}(n)),$$

the latter being precisely $\mathcal{O}(\mathcal{C}_{\textit{EXACT-MMEF}}(n))$, and hence we have the claim. □

We conclude by showing an example of execution of HEURISTIC-MMEF() on the formula:

$$\varphi = (\Box(p \to (r \wedge r))) \wedge ((q \wedge (r \wedge q))) \wedge (((\Box q \wedge \Diamond\top) \to \Diamond q) \wedge \Box(p \to (r \wedge r))),$$

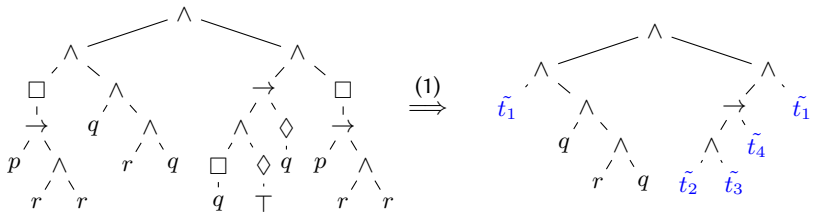
shown in Fig. 1. Upon calling HEURISTIC-MMEF($\varphi, 8, 0, 0$), after the first substitution step, we obtain a propositional formulas as in Fig. 1a, middle, with a corresponding hash table as in Fig. 1a, right; observe that, in the figure, we do not show the negations of formulae in H, which are nonetheless present. Step 1 of the execution consists of substituting all maximally modal sub-formulae of φ with fresh propositional letter; since $|\varphi| = 28$, no exact modal minimization is performed at this point. In Step 2, shown in Fig. 1b, a heuristic propositional minimization is performed because $\beta = 0$, which results in the elimination of one redundant occurrence of $\tilde{t}_1$ (in this case, however, an exact propositional minimization would not have had a different result). Even after Step 3, that is, the reverse substitution of the maximally modal formulae found in Step 1, shown in Fig. 1c, the hash table H remains the same. At this point, HEURISTIC-MMEF() is called, recursively, on the sub-formula rooted at $\wedge_1$, with no changes, on the sub-formula at $\Box_2$, which is minimized exactly, on the sub-formula rooted at $\wedge_4$, with no result, and on the sub-formula rooted at $\to_5$, again, for an exact minimization; this process is shown in Fig. 1d.

Observe that calling HEURISTIC-MMEF() on φ with a lower α would lead to the result in Fig. 1d, left, which is still smaller than the original one, but not as small as the result of the call with $\alpha = 8$.

5. Conclusions

Formula minimization is a well-known problem in the case of Boolean logic, with several applications not only in field of circuit design but also in the area of explainable artificial intelligence. In this paper we defined the same problem for the harder case of modal logic, motivated not only by the naturalness of the question but also by the fact that modal logic is now being applied to learning from non-tabular data, leading to the need of simplifying potentially very long modal formulas in order to ensure their intepretability.

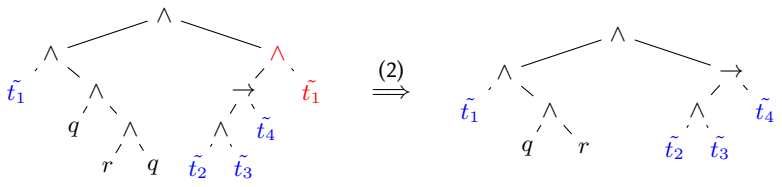
We proved that the decision problem that corresponds to minimization in the modal case is PSPACE-complete, and we proposed a heuristic algorithm, namely HEURISTIC-MMEF(), for it; we proved the soundness of our proposal, and we discussed its theoretical complexity.



H

key	value
$\Box(p \to (r \wedge r))$	$\tilde{t_1}$
$\Box q$	$\tilde{t_2}$
$\Diamond \top$	$\tilde{t_3}$
$\Diamond q$	$\tilde{t_4}$

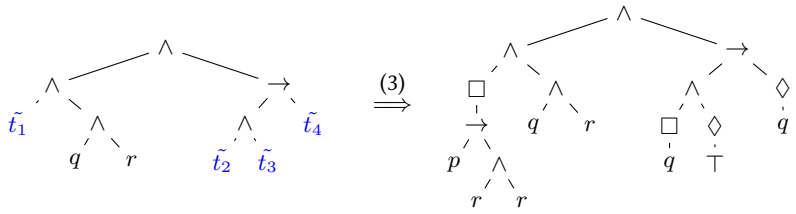
(a) In Step 1, since $|\varphi| > \alpha$ a call to PROPREPLACE() is made and every maximally modal subformula, starting from the root, is substituted with a fresh propositional letter.



H

key	value
$\Box(p \to (r \wedge r))$	$\tilde{t_1}$
$\Box q$	$\tilde{t_2}$
$\Diamond \top$	$\tilde{t_3}$
$\Diamond q$	$\tilde{t_4}$

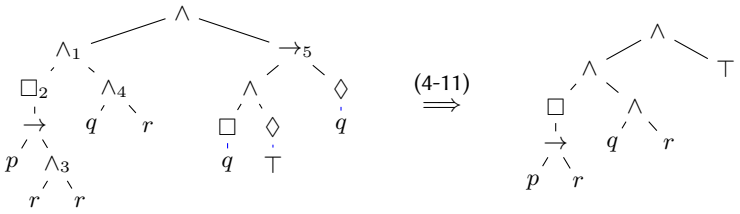
(b) In Step 2, since $|\varphi| > \beta$, a call to HEURISTIC-PMEF() is made resulting in the elimination of t_1 from the rightmost conjunct.



H

key	value
$\Box(p \to (r \wedge r))$	$\tilde{t_1}$
$\Box q$	$\tilde{t_2}$
$\Diamond \top$	$\tilde{t_3}$
$\Diamond q$	$\tilde{t_4}$

(c) In Step (3) a call to PROPREPLACE() operates the back substitution.



$H_{\wedge_4}$

key	value
$\Box(p \to (r \wedge r))$	$\tilde{t_1}$

(d) From Step 4 to Step 11 several recursive calls are performed. In particular, from Step 4 to Step 7 the formula rooted at $\wedge_1$ is minimized, but no changes occur. Step (8) corresponds to a call to EXACT-MMEF() on the formula rooted at $\Box_2$ (since its length is less than α), resulting into the elimination of the redundant conjunction $r \wedge r$. In Step 9 and 10 a recursive call on the formula rooted at $\wedge_4$ is performed, with no result. Finally, in Step (11) the procedure calls itself on the formula rooted at $\to_5$; since the length of this formula is less than α, again, this call corresponds to an execution of EXACT-MMEF(), leading to the final formula. The hash table shown at this point is precisely the one built within the recursive call on the formula rooted at $\wedge_1$. Nodes in formula syntax trees are numbered following the recursive call at which they are considered; similarly, H_4 indicates the hash table at the fourth recursive call, that is, the last one in which a modification occurs.

Figure 1: Example of execution of HEURISTIC-MMEF().

The natural continuation of this work is to implement HEURISTIC-MMEF() and to perform a comprehensive test to assess its practical usefulness. Moreover, as it is customary with heuristic solutions, some different choices may lead to different behaviours in terms of practical efficiency; we plan to explore several natural combinations and potential improvements of our proposals towards an optimal solution. Finally, it would be interesting to establish to what extent the properties of specific modal logics may improve the behaviour of HEURISTIC-MMEF() in specific cases, such as temporal or spatial logics.

This work is part of a long-term open-source project for symbolic learning and reasoning with modal, temporal, and spatial logic, written in Julia language [31].

Acknowledgments

This research has been partially funded by the FIRD project *Modal Geometric Symbolic Learning* (University of Ferrara), and the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM) project *Symbolic and Numerical Analysis of Cyber-Physical Systems* (GNCS-INdAM). Giovanni Pagliarini, Guido Sciavicco, and Ionel Eduard Stan are GNCS-INdAM members.

References

[1] C. E. Shannon, A symbolic analysis of relay and switching circuits, Transactions of the American Institute of Electrical Engineers (1938).
[2] W. V. Quine, The problem of simplifying truth functions, The American Mathematical Monthly (1952).
[3] C. Umans, The minimum equivalent dnf problem and shortest implicants, Journal of Computer and System Sciences (2001).
[4] A. R. Meyer, L. J. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential space, in: Proc. of the 13th Annual Symposium on Switching and Automata Theory, 1972, pp. 125–129.
[5] R. Rudell, A. Sangiovanni-Vincentelli, Multiple-valued minimization for pla optimization, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 6 (1987) 727–750.
[6] O. Coudert, Doing two-level logic minimization 100 times faster, in: Proc. of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, 1995, pp. 112–121.
[7] S. Hong, R. Cain, D. Ostapko, Mini: A heuristic approach for logic minimization, IBM Journal of Research and Development 18 (1974) 443 – 458.
[8] J. Hlavicka, P. Fiser, BOOM - A heuristic boolean minimizer, Computing and Informatics 22 (2003) 19–51.
[9] E. Calò, J. Levy, General boolean formula minimization with QBF solvers, in: Proc. of the 25th International Conference of the Catalan Association for Artificial Intelligence, volume 375 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 347–358.
[10] J. Marques-Silva, Logic-based explainability in machine learning, in: Proc. of the 18th International Summer School 2022 on Reasoning Web. Causality, Explanations and Declarative Knowledge, volume 13759 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 24–104.
[11] T. Lampert, Minimizing disjunctive normal forms of pure first-order logic, Logic Journal of the IGPL 25 (2017) 325–347.
[12] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, volume 53 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 2001.
[13] M. Ben-Ari, Z. Manna, A. Pnueli, The temporal logic of branching time, in: Proc. of the 8th Annual ACM Symposium on Principles of Programming Languages, ACM Press, 1981, pp. 164–176.
[14] J. Halpern, Y. Shoham, A propositional modal logic of time intervals, J. ACM 38 (1991) 935–962.
[15] A. Pnueli, The temporal logic of programs, in: Proc. of the 18th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1977, pp. 46–57.
[16] M. Aiello, J. van Benthem, A modal walk through space, Journal of Applied Non Classical Logics 12 (2002) 319–364.
[17] L. Sikos, Description Logics in Multimedia Reasoning, Springer, 2017.
[18] G. Bechini, E. Losi, L. Manservigi, G. Pagliarini, G. Sciavicco, I. E. Stan, M. Venturini, Statistical rule extraction for gas turbine trip prediction, Journal of Engineering for Gas Turbines and Power 145 (2023) GTP–22–1408.
[19] F. Manzella, G. Pagliarini, G. Sciavicco, I. Stan, The voice of COVID-19: breath and cough recording

classification with temporal decision trees and random forests, Artificial Intelligence in Medicine 137 (2023) 102486.
[20] G. Pagliarini, G. Sciavicco, Interpretable land cover classification with modal decision trees, European Journal of Remote Sensing 56 (2023) 2262738.
[21] S. Mazzacane, M. Coccagna, F. Manzella, G. Pagliarini, V. Sironi, A. Gatti, E. Caselli, G. Sciavicco, Towards an objective theory of subjective liking: A first step in understanding the sense of beauty, PLOS ONE (2023) 1–20.
[22] D. Della Monica, G. Pagliarini, G. Sciavicco, I. Stan, Decision trees with a modal flavor, in: Proc. of the 21st International Conference of the Italian Association for Artificial Intelligence, volume 13796 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 47–59.
[23] R. E. Ladner, The computational complexity of provability in systems of modal propositional logic, SIAM Journal on Computing (1977).
[24] M. Fitting, Tableau methods of proof for modal logics, Notre Dame Journal of Formal Logic 13 (1972) 237–247.
[25] M. F, Single step tableaux for modal logics, Journal of Automatic Reasoning 24 (2000) 319–364.
[26] F. Giunchiglia, R. Sebastiani, Building decision procedures for modal logics from propositional decision procedures: The case study of modal k(m), Information and Computation 162 (2000) 158–178.
[27] C. Areces, R. Gennari, J. Heguiabehere, M. de Rijke, Tree-based heuristics in modal theorem proving, in: Proc. of the 14th European Conference on Artificial Intelligence (ECAI), IOS Press, 2000, pp. 199–203.
[28] G. Pan, U. Sattler, M. Vardi, Bdd-based decision procedures for the modal logic K, Journal of Applied Non Classical Logics 16 (2006) 169–208.
[29] R. Sebastiani, A. Tacchella, SAT techniques for modal and description logics, in: Handbook of Satisfiability - Second Edition, volume 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 1223–1266.
[30] M. Bienvenu, Prime implicates and prime implicants: From propositional to modal logic, J. Artif. Intell. Res. (JAIR) (2009).
[31] F. Manzella, G. Pagliarini, A. Paparella, G. Sciavicco, I. Stan, Sole.jl – Symbolic Learning in Julia, https://github.com/aclai-lab/Sole.jl, 2024.