# Explainability via Short Formulas: the Case of Propositional Logic with Implementation

Reijo Jaakkola[1], Tomi Janhunen[2], Antti Kuusisto[3], Masood Feyzbakhsh Rankooh[4] and Miikka Vilander[5,*]

*Tampere University, FI-33014, Tampere University, Finland*

**Abstract**

We conceptualize explainability in terms of logic and formula size, giving a number of related definitions of explainability in a very general setting. Our main interest is the so-called special explanation problem which aims to explain the truth value of an input formula in an input model. The explanation is a formula of minimal size that (1) agrees with the input formula on the input model and (2) transmits the involved truth value to the input formula globally, i.e., on every model. As an important example case, we study propositional logic in this setting and show that the special explainability problem is complete for the second level of the polynomial hierarchy. We also provide an implementation of this problem in answer set programming and investigate its capacity in relation to explaining answers to the n-queens and dominating set problems.

**Keywords**

explainability, answer set programming, satisfiability checking, computational complexity

## 1. Introduction

This paper investigates explainability in a general setting. The key in our approach is to relate explainability to formula size. We differentiate between the *general* and *special explanation problems*. The general explanation problem for a logic L takes as input a formula $\varphi \in \mathrm{L}$ and outputs an equivalent formula of minimal size. Thus the objective is to explain the global behavior of $\varphi$. For example, we can consider $\neg\neg\neg\neg\neg\neg p$ to be globally explained by $p$. In contrast, the goal of the special explanation problem is to explicate why an input formula $\psi$ gets the truth value $b$ in an input model $M$. Given a tuple $(M, \psi, b)$, the problem outputs a formula $\chi$ of minimal size such that (1) the formula $\chi$ obtains the same truth value $b$ on $M$, and (2) on every model $M'$ where $\chi$ gets the truth value $b$, also $\psi$ gets that same truth value. Intuitively, this second condition states that the given truth value $b$ of $\chi$ on a model $M'$ causes $\psi$ to be judged similarly. In summary, the special explanation problem gives reasons why a piece of data (or a

*Accepted for presentation at the 29th RCRA Workshop, 2022, Genoa, Italy*

*Corresponding author.

reijo.jaakkola@tuni.fi (R. Jaakkola); tomi.janhunen@tuni.fi (T. Janhunen); antti.kuusisto@tuni.fi (A. Kuusisto); masood.feyzbakhshrankooh@tuni.fi (M. F. Rankooh); miikka.vilander@tuni.fi (M. Vilander)
https://reijojaakkola.github.io/ (R. Jaakkola); https://tuni.fi/en/tomi-janhunen (T. Janhunen); https://homepages.tuni.fi/antti.kuusisto/ (A. Kuusisto)
0000-0003-4714-4637 (R. Jaakkola); 0000-0002-2029-7708 (T. Janhunen); 0000-0003-1356-8749 (A. Kuusisto); 0000-0001-5660-3052 (M. F. Rankooh); 0000-0002-7301-939X (M. Vilander)

model) is treated in a given way (i.e., obtains a given truth value) by a classifier (or a formula).

The two explanation problems give rise to corresponding questions of explainability. The *general explainability problem* for a logic L asks, given a formula $\varphi \in \mathrm{L}$ and $k \in \mathbb{N}$, whether there exists a formula equivalent to $\psi$ of size at most $k$. The *special explainability problem* gets as input a tuple $(M, \psi, b, k)$, and the task is then to check whether there exists a formula $\chi$ of size at most $k$ satisfying the above conditions (1) and (2) of the special explanation problem.

As an important particular case, we study the special explainability problem of propositional logic (PL) in detail. We prove that the problem is $\Sigma_2^P$-complete. This is an important result whose usefulness lies in its implications that go far beyond PL itself. Indeed, the result gives a robust lower bound for various logics. We demonstrate this by establishing $\Sigma_2^P$-completeness of the special explainability problem of S5 modal logic; the lower bound follows *directly* from the case of PL, while the upper bound is very easy to obtain. We observe that a rather wide range of logics with satisfiability in NP have $\Sigma_2^P$-complete special explainability problems.

As a further theoretical result, we prove that, when limiting to explaining only the positive truth value $\top$, the special explainability problem is only NP-complete for *CNF*-formulas of PL. As a corollary, we get NP-completeness of the problem for *DNF*-formulas in restriction to the truth value $\bot$. While theoretically interesting, these results are also relevant from the point of view of applications, as quite often real-life classification scenarios require explanations only in the case of one truth value. For example, explanations concerning automated insurance decisions are typically relevant only in the case of rejected applications. In addition to the NP-completeness results, we also show that restricting to a single truth value here is necessary for obtaining the given complexity (supposing coNP $\nsubseteq$ NP).

We provide an implementation of the special explainability problem of PL based on *answer-set programming* (ASP) [1]. Generally, ASP is a logic programming language based on stable-model semantics and propositional syntax, particularly Horn clauses. ASP is *especially suitable* and almost custom-made for implementing the special explainability problem of PL, as ASP is designed precisely for the complexity levels up to $\Sigma_2^P$. Indeed, while the disjunction-free fragments of ASP [2] cover the first level of the polynomial hierarchy in a natural way, proper disjunctive rules with cyclic dependencies become necessary to reach the second one [3].

We test the implementation via experiments with benchmarks based on the *n-queens* and *dominating set problems*. The experiments provide concrete and compact explanations why a particular configuration of queens on the generalized chessboard or a particular set of vertices of a graph is or is not an acceptable solution to the involved problem. Runtimes scale exponentially in the size of the instance and negative explanations tend to be harder to compute than positive ones. Also, we observe from the experiments that the respective optimization variants of explanations problems can be computationally more effective, leaving the exact bound on the size of explaining formulas open and relieving the user from providing a particular bound.

Concerning related work, while the literature on explainability is extensive, only a small part of it is primarily based on logic. In relation to the current paper, the special explainability problem in the particular case of PL has some similarities with the *prime implicant* (or PI) explanations of [4]. However, there are some key differences. PI-explanations are defined in terms of finding a minimal subset of features—or propositions—that suffice to explain the input instance of a Boolean decision function. Our definition of special explainability allows for any kind of formula as output. In the propositional case, we separately prove that a subconjunction of the literals in the input is always a possible output. We end up with a similar goal of removing propositions, but from a different starting point and in a different setting. In the case of other logics, such as FO, the resemblance to prime implicant explanations decreases. Concerning results, [4] does not provide a full complexity analysis relating to the studied problems. Although PI-explanations are defined for any decision function, the algorithms used in [4] to compute PI-explanations have ordered binary decision diagrams (OBDD) as inputs and outputs. For these algorithms, the authors give empirical runtimes but no complexity bounds. Thus, in our work, while the space of potential input and output formulas is different, we also give a complete complexity analysis of the special explainability problem in addition to experiments.

In [5], Umans shows that the *shortest implicant problem* is $\Sigma_2^p$-complete, thereby solving a long-standing open problem of Stockmeyer. An implicant of $\varphi$ is a conjunction $\chi$ of literals such that $\chi \vDash \varphi$. The shortest implicant problem asks whether there is an implicant of $\varphi$ with size at most $k$. Size is defined as the number of occurrences of literals. Below we prove that the special explainability problem for PL can be reduced to certain particular implicant problems. However, despite this, the work of Umans does not directly modify to give the $\Sigma_2^p$-completeness result of the special explainability problem. The key issue is that the explainability problem requires an interpolant between a set $\chi$ of literals and a formula $\varphi$, where $\chi$ has precisely the same set of propositions symbols as $\varphi$. Thus we need to give an independent proof for the $\Sigma_2^P$ lower bound. Also, formula size in [5] is measured in a more coarse way.

## 2. Preliminaries

Let $\Phi$ be a set of proposition symbols. The set $\mathrm{PL}(\Phi)$ of formulas of **propositional logic** PL over $\Phi$ is given by the grammar $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi)$, where $p \in \Phi$. **Literals** are either atoms $p$ or their negations $\neg p$, also known as **positive** and **negative** literals, respectively. A $\Phi$-**assignment** is a function $s : \Phi \to \{0, 1\}$. When $\Phi$ is clear from the context or irrelevant, we simply refer to assignments rather than $\Phi$-assignments. We define the semantics of propositional logic in the usual way, and we write $s \vDash \varphi$ if the assignment $s$ **satisfies** the formula $\varphi \in \mathrm{PL}(\Phi)$. Alternatively, we can use the **standard valuation** function $v_\Phi$ defined such that $v_\Phi(s, \varphi) = 1$ if $s \vDash \varphi$ and otherwise $v_\Phi(s, \varphi) = 0$. Below we sometimes use the set $\{\bot, \top\}$ instead of $\{0, 1\}$,

A formula $\psi \in \mathrm{PL}(\Phi)$ is a **logical consequence** of $\varphi \in \mathrm{PL}(\Phi)$, denoted $\varphi \vDash \psi$, if for every $\Phi$-assignment $s$, $s \vDash \varphi$ implies $s \vDash \psi$. A formula $\chi \in \mathrm{PL}(\Phi)$ is an **interpolant** between $\varphi$ and $\psi$ if $\varphi \vDash \chi$ and $\chi \vDash \psi$. For a finite $\Phi$, we say that a formula $\varphi$ is a **maximal conjunction** w.r.t. $\Phi$ if $\varphi$ is a conjunction of exactly one *literal* for each $p \in \Phi$. *A $\Phi$-assignment $s$ can be naturally identified with a maximal conjunction,* for example $\{(p, 1), (q, 0)\}$ identifies with $p \wedge \neg q$. A formula $\chi$ is a **subconjunction** of $\varphi$ if $\chi$ is a conjunction of literals occurring in $\varphi$. The **size** of $\varphi$, denoted *size*$(\varphi)$, is the number of occurrences of proposition symbols, binary connectives and negations in $\varphi$. For example, the size of $\neg\neg(p \wedge p)$ is 5 as it has one $\wedge$ and two occurrences of both $\neg$ and $p$.

## 3. Notions of explanation and explainability

In this section we introduce four natural problems concerning the general and special perspectives to explainability. The general problems deal with the question of explaining the entire behavior of a classifier, whereas the special ones attempt to explicate why a single input instance was classified in a given way. We give very general definitions of these problems, and for that we will devise a very general definition of the notion of a logic. Our definition of a logic covers various kinds of classifiers in addition to standard formal logics, including logic programs, Turing machines, neural network models, automata, and the like.

**Definition 1.** A **logic** is a tuple $(\mathcal{M}, \mathcal{F}, v, m)$ where $\mathcal{M}$ and $\mathcal{F}$ are sets; $v : \mathcal{M} \times \mathcal{F} \to V$ is a function mapping to some set $V$; and $m : \mathcal{F} \to \mathbb{N}$ is a function. Emphasizing the set $V$, we can also call $(\mathcal{M}, \mathcal{F}, v, m)$ a $V$-**valued logic**.

Intuitively, we can think of $\mathcal{M}$ as a set of models and $\mathcal{F}$ as a set of formulas. The function $v : \mathcal{M} \times \mathcal{F} \to V$ gives the semantics of the logic, with $v(\mathfrak{M}, \varphi)$ being the truth value of $\varphi$ in $\mathfrak{M}$. We call $v$ a **valuation**. The function $m$ gives a complexity measure for the formulas in $\mathcal{F}$, such as, for example, formula size.

**Example 2.** Propositional logic PL over a set $\Phi$ of proposition symbols can be defined as a tuple $(\mathcal{M}, \mathcal{F}, v, m)$, where $\mathcal{M}$ is the set of $\Phi$-assignments; $\mathcal{F}$ the set $\mathrm{PL}(\Phi)$ of formulas; $v : \mathcal{M} \times \mathcal{F} \to \{0, 1\}$ is the standard valuation $v_\Phi$; and $m(\varphi) = \mathit{size}(\varphi)$ for all $\varphi \in \mathcal{F}$.

Now, the following example demonstrates that we can consider much more general scenarios than ones involving the standard formal logics.

**Example 3.** Let $\mathcal{M}$ be a set of data and $\mathcal{F}$ a set of programs for classifying the data, that is, programs that take elements of $\mathcal{M}$ as inputs and output a value in some set $V$ of suitable outputs. Now $v : \mathcal{M} \times \mathcal{F} \to V$ is just the function such that $v(D, P)$ is the output of $P \in \mathcal{F}$ on the input $D \in \mathcal{M}$. The function $m$ can quite naturally give the program size for each $P \in \mathcal{F}$. If we redefine the domain of $m$ to be $\mathcal{M} \times \mathcal{F}$, we can let $m(D, P)$ be for example the running time of the program $P$ on the input $D$, or the length of the computation (or derivation) table.

Given a logic, we define the equivalence relation $\equiv\, \subseteq \mathcal{F} \times \mathcal{F}$ such that $(M, M') \in\, \equiv$ if and only if $v(M, \varphi) = v(M', \varphi)$ for all $\varphi \in \mathcal{F}$. We shall now define four formal problems relating to explainability. The problems do work especially well for finite $V$, but this is not required as long as the elements of $V$ are representable in the sense that they can be used as inputs to computational problems.

**Definition 4.** Let $L = (\mathcal{M}, \mathcal{F}, v, m)$ be a logic. We define the following four problems for $L$.
**General explanation problem:**
*Input*: $\varphi \in \mathcal{F}$, *Output*: $\psi \in \mathcal{F}$
*Description*: Find $\psi \in \mathcal{F}$ with $\psi \equiv \varphi$ and minimal $m(\psi)$.

**Special explanation problem**
*Input*: $(\mathfrak{M}, \varphi, b)$ where $\mathfrak{M} \in \mathcal{M}$, $\varphi \in \mathcal{F}$ and $b \in V$, *Output*: $\psi \in \mathcal{F}$ or error
*Description*: If $v(\mathfrak{M}, \varphi) \neq b$, output error. Else find $\psi \in \mathcal{F}$ with minimal $m(\psi)$ such that the following two conditions hold:
(1) $v(\mathfrak{M}, \psi) = b$ and
(2) For all $\mathfrak{M}' \in \mathcal{M}$, $v(\mathfrak{M}', \psi) = b \Rightarrow v(\mathfrak{M}', \varphi) = b$.

**General explainability problem**
*Input*: $(\varphi, k)$, where $\varphi \in \mathcal{F}$ and $k \in \mathbb{N}$, *Output*: Yes or no
*Description*: If there is $\psi \in \mathcal{F}$ with $\psi \equiv \varphi$ and $m(\psi) \leq k$, output yes. Otherwise output no.

**Special explainability problem**
*Input*: $(\mathfrak{M}, \varphi, b, k)$ where $\mathfrak{M} \in \mathcal{M}$, $\varphi \in \mathcal{F}$, $b \in V$ and $k \in \mathbb{N}$, *Output*: Yes or no
*Description*: Output "yes" if and only if there exists some $\psi \in \mathcal{F}$ with $m(\varphi) \leq k$ such that the conditions (1) and (2) of the special explanation problem hold.

In the full version, we shall generalize these definitions, but the current ones suffice here. However, already in the current framework, the notions are quite flexible. Notice, for example, that while the set $\mathcal{M}$ may typically be considered a set of models, or pieces of data, there are many further natural possibilities. For instance, $\mathcal{M}$ can be a set of formulas. This nicely covers, e.g., model-free settings based on proof systems.

## 3.1. Special explainability for PL

Let $(\varphi, \psi, b)$ be an input to the special explanation problem of propositional logic, where $\varphi$ is a maximal conjunction w.r.t. some (any) finite $\Phi$ (thus encoding a $\Phi$-assignment) and $\psi$ a $\Phi$-formula. The special explanation problem can be reformulated equivalently in the following way. (1) Suppose $b = \top$. If $\varphi \vDash \psi$, find a minimal interpolant between $\varphi$ and $\psi$. Else output `error`. (2) Suppose $b = \bot$. If $\varphi \vDash \neg\psi$, find a minimal interpolant between $\psi$ and $\neg\varphi$. Else output `error`.

Let $\varphi \in \mathrm{PL}(\Phi)$ be a conjunction of literals. Let $P(\varphi)$ and $N(\varphi)$ be the sets of positive and negative literals in $\varphi$, respectively. We denote the De Morgan transformations of $\varphi$ and $\neg\varphi$ by

$$\mathrm{DM}(\varphi) := \bigwedge_{p \in P(\varphi)} p \wedge \neg\Big(\bigvee_{\neg q \in N(\varphi)} q\Big) \quad \text{and} \quad \mathrm{DM}(\neg\varphi) := \neg\Big(\bigwedge_{p \in P(\varphi)} p\Big) \vee \bigvee_{\neg q \in N(\varphi)} q.$$

**Lemma 5.** *Let $\Phi$ be a finite set of proposition symbols, let $\varphi$ be a maximal conjunction w.r.t. $\Phi$ and let $\psi \in \mathrm{PL}(\Phi)$. (1) If $\varphi \vDash \psi$, then there is a subconjunction $\chi$ of $\varphi$ such that $\mathrm{DM}(\chi)$ is a minimal interpolant between $\varphi$ and $\psi$. (2) If $\varphi \vDash \neg\psi$, then there is a subconjunction $\chi$ of $\varphi$ such that $\mathrm{DM}(\neg\chi)$ is a minimal interpolant between $\psi$ and $\neg\varphi$.*

*Proof.* Assume that $\varphi \vDash \psi$. Clearly at least one minimal interpolant exists, as for example $\varphi$ itself is an interpolant. Let $\theta$ be a minimal interpolant. Let $\Phi(\theta)$ be the set of proposition symbols occurring in $\theta$. We transform $\theta$ into an equivalent formula $\theta'$ in $\Phi(\theta)$-full disjunctive normal form where each disjunct is a maximal conjunction w.r.t. $\Phi(\theta)$. To see that such a form always exists, first consider the DNF of $\theta$ and then, for each disjunct, fill in all possible values of any missing propositions (thus possibly increasing the number of disjuncts).

Now, as $\varphi$ is a maximal conjunction w.r.t. $\Phi$, exactly one disjunct of $\theta'$ is a subconjunction of $\varphi$. Let $\chi$ denote this disjunct. As $\varphi \vDash \psi$, the formula $\chi$ is clearly an interpolant between $\varphi$ and $\psi$.

Now, each proposition in $\Phi(\theta)$ occurs in $\chi$ and thus also in $\mathrm{DM}(\chi)$ exactly once, so $\mathrm{DM}(\chi)$ has at most the same number of occurrences of proposition symbols and binary connectives as $\theta$. Furthermore, $\mathrm{DM}(\chi)$ has at most one negation. If $\theta$ has no negations, then we claim $\mathrm{DM}(\chi)$ also has none. To see this, note that $\chi$ is a disjunct of $\theta'$ and $\theta'$ is equivalent to $\theta$, so $\chi \vDash \theta$. As $\theta$ is negation-free, we have $\chi' \vDash \theta$ where $\chi'$ is obtained from $\chi$ by removing all the negative literals. Hence, by the minimality of $\theta$, we have $\chi' = \chi$. Thus $\chi$ and $\mathrm{DM}(\chi)$ are negation-free. We have shown that $size(\mathrm{DM}(\chi)) \leq size(\theta)$, so $\mathrm{DM}(\chi)$ is a minimal interpolant.

Suppose then that $\varphi \vDash \neg\psi$. Let $\theta$ be a minimal interpolant between $\psi$ and $\neg\varphi$. In a dual fashion compared to the positive case above, we transform $\theta$ into an equivalent formula $\theta'$ in $\Phi(\theta)$-full conjunctive normal form. Additionally let $\varphi'$ be the negation normal form of $\neg\varphi$. Now $\varphi'$ is a disjunction of literals and exactly one conjunct of $\theta'$ is a subdisjunction of $\varphi'$. Let $\chi'$ denote this conjunct and let $\chi$ denote the negation normal form of $\neg\chi'$. Now $\chi$ is a subconjunction of $\varphi$ and $\neg\chi$ is an interpolant between $\psi$ and $\neg\varphi$.

As in the positive case, $\mathrm{DM}(\neg\chi)$ has at most the same number of proposition symbols and binary connectives as the minimal interpolant $\theta$. The formula $\mathrm{DM}(\neg\chi)$ again has at most one negation so we only check the case where $\theta$ has no negations. Recall that $\neg\chi$ is equivalent to $\chi'$, which in turn is a conjunct of $\theta'$. Thus $\theta \vDash \chi'$. As $\theta$ has no negations and $\chi'$ is a disjunction of literals, we have $\theta \vDash \chi''$, where $\chi''$ is obtained from $\chi'$ by removing all the negative literals. By the minimality of $\theta$ we obtain $\chi'' = \chi'$ so $\chi'$ has no negations. Thus also $\mathrm{DM}(\neg\chi)$ is negation-free and is a minimal interpolant. $\square$

The above lemma implies that for propositional logic, it suffices to consider subconjunctions of the input in the special explanation and explainability problems. This will be very useful both in the below theoretical considerations and in implementations.

We next prove $\Sigma_2^p$-completeness of the special explainability problem via a reduction from $\Sigma_2 SAT$, which is well-known to be $\Sigma_2^p$-complete. The input of the problem is a quantified Boolean formula $\varphi$ of the form $\exists p_1 \dots \exists p_n \forall q_1 \dots \forall q_m \theta(p_1, \dots, p_n, q_1, \dots, q_m)$. The output is yes iff $\varphi$ is true.

**Theorem 6.** *The special explainability problem for* PL *is* $\Sigma_2^p$*-complete.*

*Proof.* The upper bound is clear. For the lower bound, we will give a polynomial time (Karp-) reduction from $\Sigma_2 SAT$. Consider an instance $\exists p_1 \dots \exists p_n \forall q_1 \dots \forall q_m \theta(p_1, \dots, p_n, q_1, \dots, q_m)$ of $\Sigma_2 SAT$. We start by introducing, for every existentially quantified Boolean variable $p_i$, a new proposition symbol $\overline{p}_i$. Denoting $\theta(p_1, \dots, p_n, q_1, \dots, q_m)$ simply by $\theta$, we define

$$\psi := \bigwedge_{i=1}^{n} (p_i \vee \overline{p}_i) \wedge \Big(\theta \vee \bigvee_{i=1}^{n} (p_i \wedge \overline{p}_i)\Big).$$

We let $s$ be the valuation mapping all proposition symbols to 1, i.e., the assignment corresponding to the maximal conjunction $\varphi_s := \bigwedge_{i=1}^{n} (p_i \wedge \overline{p}_i) \wedge \bigwedge_{j=1}^{m} q_j$ w.r.t. the set of proposition symbols in $\psi$. Clearly $\varphi_s \vDash \psi$. We now claim that there exists an interpolant of size at most $2n-1$ between $\varphi_s$ and $\psi$ iff the original instance of $\Sigma_2 SAT$ is true.

Suppose first that the original instance of $\Sigma_2 SAT$ is true. Thus there exists a tuple $(p_1, \dots, p_n) \in \{0,1\}^n$ such that $\forall q_1 \dots q_m \theta(p_1, \dots, p_n, q_1, \dots, q_m)$ is true. Consider now the subconjunction $\chi := \bigwedge_{s(p_i)=1} p_i \;\wedge\; \bigwedge_{s(p_i)=0} \overline{p}_i$ of $\varphi_s$. Clearly $\chi$ is of size $2n-1$. It is easy to see that $\chi$ is also an interpolant between $\varphi_s$ and $\psi$.

Suppose then that $\chi$ is an interpolant of size at most $2n-1$ between $\varphi_s$ and $\psi$. Using Lemma 5, we can assume that $\chi$ is a subconjunction of $\varphi_s$. Since $\chi$ has size at most $2n-1$, it can contain at most $n$ proposition symbols. Furthermore, $\chi$ must contain, for every $i \in \{1, \dots, n\}$, either $p_i$ or $\overline{p}_i$, since otherwise $\chi$ would not entail $\bigwedge_{i=1}^{n} (p_i \vee \overline{p}_i)$. Thus $\chi$ contains precisely $n$ proposition symbols. More specifically, $\chi$ contains, for every $i \in \{1, \dots, n\}$, either $p_i$ or $\overline{p}_i$. Now, we define a tuple $(u_1, \dots, u_n) \in \{0, 1\}^n$ by setting $u_i = 1$ if $\chi$ contains $p_i$ and $u_i = 0$ if $\chi$ contains $\overline{p}_i$. It is easy to see that $\forall q_1 \dots q_m \theta(u_1, \dots, u_n, q_1, \dots, q_m)$ is true. □

The above theorem immediately implies a wide range of corollaries. Recall that S5 is the system of modal logic where the accessibility relations are equivalences. The special explainability problem for S5 has as input a pointed S5-model $(\mathfrak{M}, w)$, an S5-formula $\varphi$, $b \in \{\top, \bot\}$ and $k \in \mathbb{N}$.

**Corollary 7.** *The special explainability problem for* S5 *is* $\Sigma_2^p$*-complete.*

*Proof.* The lower bound follows immediately from Theorem 6, while the upper bound follows from the well-known fact that the validity problem for S5 is coNP-complete [6]. □

Note indeed that the $\Sigma_2^p$ lower bound for propositional logic is a rather useful result, implying $\Sigma_2^p$-completeness of special explainability for various logics with an NP-complete satisfiability.

We next show that if the formula $\psi$ in the special explainability problem is restricted to *CNF*-formulas and we consider only the case $b = \top$, then the problem is NP-complete. The following example demonstrates that it is necessary to restrict to the case $b = \top$.

**Example 8.** Let $\psi \in \mathrm{PL}(\Phi)$ be an arbitrary *CNF*-formula and let $q$ be a proposition symbol such that $q \notin \Phi$. Suppose that $s \not\models \psi$, where $s(p) = 1$ for every $p \in \Phi$. Consider now the formula $\varphi_q := q \vee \bigvee_{p \in \Phi} \neg p$. Note that $\neg\varphi_q$ is equivalent to a maximal conjunction w.r.t. $\Phi \cup \{q\}$. Clearly $\psi \wedge \neg q \models \varphi_q$, since $\psi$ entails that $\bigvee_{p \in \Phi} \neg p$. We now claim that there exists an interpolant $\chi$ of size one between $\psi \wedge \neg q$ and $\varphi_q$ iff $\psi$ is unsatisfiable. First, we note that the only proposition from $\Phi \cup \{q\}$ which entails $\varphi_q$ is $q$. But the only way $q$ can be an interpolant between $\psi \wedge \neg q$ and $\varphi_q$ is that $\psi$ is unsatisfiable. Conversely, if $\psi$ is unsatisfiable, then $q$ is clearly an interpolant between $\psi \wedge \neg q$ and $\varphi_q$. Since the unsatisfiability problem of *CNF*-formulas is coNP-hard, the special explainability problem for *CNF*-formulas is also coNP-hard.

To prove NP-hardness, we will give a reduction from the dominating set problem. For a graph $G = (V, E)$, a **dominating set** $D \subseteq V$ is a set of vertices such that every vertex not in $D$ is adjacent to a vertex in $D$. The input of the dominating set problem is a graph and a natural number $k$. The output is yes, if the graph has a dominating set of at most $k$ vertices.

**Theorem 9.** *For CNF-formulas, the special explainability problem with* $b = \top$ *is NP-complete. The lower bound holds even if we restrict our attention to formulas without negations.*

*Proof.* For the upper bound, let $\psi \in \mathrm{PL}(\Phi)$ be a *CNF*-formula and let $\varphi$ be a maximal conjunction w.r.t. $\Phi$. We want to determine whether there is an interpolant of size at most $k$. Using Lemma 5, it suffices to determine whether there is a subconjunction $\chi$ of $\varphi$ such that $\mathit{size}(\mathrm{DM}(\chi)) \leq k$.

Our nondeterministic procedure will start by guessing a subconjunction $\chi$ of $\varphi$. If $\mathit{size}(\mathrm{DM}(\chi)) > k$, then it rejects. Otherwise we replace the formula $\psi$ with the formula $\psi'$ which is obtained from $\psi$ by replacing each proposition symbol $p$ that occurs in $\chi$ with either $\top$ or $\bot$, depending on whether $p$ occurs positively or negatively in $\chi$. Now, if $\psi'$ is valid, then our procedure accepts, and if it is not, then it rejects. Since the validity of *CNF*-formulas can be decided in polynomial time, our procedure runs in polynomial time as well.

For the lower bound we will give a reduction from the dominating set problem. Consider a graph $G = (V, E)$ and a parameter $k$. Let

$$\psi := \bigwedge_{v \in V} \Big( p_v \vee \bigvee_{(v,u) \in E} p_u \Big) \tag{1}$$

and $\varphi := \bigwedge_{v \in V} p_v$. Now $\varphi \models \psi$ and $\psi$ is a *CNF*-formula. It is easy to verify that there exists an interpolant $\theta$ of size at most $2k - 1$ if and only if $G$ has a dominating set of size at most $k$. $\square$

By simply negating formulas, we obtain that the special explainability problem with $b = \bot$ is NP-complete for *DNF*-formulas (and in general coNP-hard, see Example 8).

**Corollary 10.** *For DNF-formulas, the special explainability problem with $b = \bot$ is NP-complete.*

## 3.2. On general explainability

The general explainability problem for propositional logic has been discussed in the literature under motivations unrelated to explainability. The **minimum equivalent expression problem** MEE asks, given a formula $\varphi$ and an integer $k$, if there exists a formula equivalent to $\varphi$ and of size at most $k$. This problem has been shown in [7] to be $\Sigma_2^p$-complete under Turing reductions, with formula size defined as the number of occurrences of proposition symbols and with formulas in negation normal form. The case of standard reductions is open.

For logics beyond PL, the literature on the complexity of formula minimization is surprisingly scarce. The study of formula size in first-order and modal logics has mainly focused on particular properties that either can be expressed very succinctly or via a very large formulas. This leads to relative succinctness results between logics. For lack of space, we shall not discuss the general explainability problem further in the current article, but instead leave the topic for the future.

## 4. Implementation

In this section, we devise a proof-of-concept implementation of explainability problems defined above. The implementation exploits the ASP fragment of the *Clingo* system[1] combining the *Gringo* grounder with the *Clasp* solver. However, we adopt the modular approach of [8] for the representation of oracles, thus hiding disjunctive rules and their saturation from encodings. Since *CNF*-formulas are dominant in the context of SAT checking, we devise our first implementation under an assumption that input formulas take this form. Thus, in spirit of Lemma 5, $\varphi$ is essentially a set $L$ of literals and $\psi$ is a set $S$ of *clauses*, i.e., disjunctions of literals. To enable meta-level encodings in ASP, a *CNF*-formula in DIMACS format can be reified into a set of first-order (ground) facts using the *lpreify* tool[2] (option flag `-d`). Using additional rules, we define domain predicates `clause/1`, `pcond/2`, and `ncond/2` for identifying clauses, their positive conditions, and negative conditions respectively. The literals in the set $L$ are expressed by using domain predicates `plit/1` and `nlit/1` for positive and negative literals, respectively.

[1]https://potassco.org/clingo/
[2]https://github.com/asptools/software

Listing 1: Checking Positive Precondition (Lemma 5)

```
1 :- clause(C), nlit(P): pcond(C,P); plit(N): ncond(C,N).
2 simp(C) :- plit(P), pcond(C,P).         simp(C) :- nlit(N), ncond(C,N).
3 simp(C) :- pcond(C,A), ncond(C,A).
4 :- clause(C), not simp(C), pcond(C,P), not plit(P), not nlit(P).
5 :- clause(C), not simp(C), ncond(C,N), not plit(N), not nlit(N).
```

Listing 2: Checking Negative Precondition (Lemma 5)

```
1 t(A) :- plit(A), atom(A).
2 { t(A) } :- atom(A), not plit(A), not nlit(A).
3 :- clause(C), not t(P): pcond(C,P); t(N): ncond(C,N).
```

We relax the requirement that the set of literals $L$ is maximal, so that any three-valued interpretation can be represented. However, the precondition for the positive (resp. negative) explanation is essentially the same: the result $S|_L$ of *partially evaluating* $S$ with respect to $L$ must remain valid (resp. unsatisfiable) in accordance to Lemma 5. The positive check is formalized in Listing 1. The constraint in Line 1 excludes the possibility that $L$ falsifies $S$ directly. Lines 2–3 detect which clauses of $S$ are immediately true given $L$ and removed from $S|_L$ altogether. Rules in Lines 4 and 5 deny any clause containing yet open literals that could be used to falsify the clause in question. The net effect is that the encoding extended by facts describing $L$ and $S$ has an answer set iff $S|_L$ is valid. Since the scope of negation is restricted to domain predicates only, the check is effectively polytime. The negative case can be handled by a single ASP program evaluating a coNP query, see Listing 2. The rule in Line 1 infers any positive literal in $L$ to be true while the negative ones in $L$ remain false *by default*. In Line 2, the truth values of atoms undefined in $L$ are freely chosen. The constraint in Line 3 ensures that each clause in the input $S$ must be satisfied. Thus $S|_L$ is unsatisfiable iff the encoding extended by facts describing $L$ and $S$ has no answer set. In general, this check is deemed worst-case exponential, but for maximal $L$, the task reduces to simple polytime propagation as no choices are active in Line 2.

Our more general goal is to find *minimum-size* explanations $L' \subseteq L$ possessing the identical property as required from $L$, i.e., $S|_{L'}$ is either valid or unsatisfiable. As regards the size of $L'$, we leave the mapping back to a minimum-size formula as a post-processing step. In the negative case (the second item of Lemma 5), the idea is formalized by Listing 3. While the (fixed) set $L$ is specified as before using predicates `plit/1` and `nlit/1`, the subset $L'$ is determined by choosing $L$-compatible truth values for atoms in Line 1. The resulting size of $L'$ is put subject to minimization in Line 3 if `k=0`, as set by default in Line 2. Positive values `k>0` set by the user activate the *special explainability* mode: the size of $L'$ is at most `k` by the cardinality constraint in Line 4. Besides this objective, we check that $L' \cup S$ is unsatisfiable by using an oracle encoded in Listing 4. The *input atoms* (cf. [8]) are declared in Line 1. The predicate `et/1` captures a two-valued truth assignment compatible with $L'$ as enforced by Lines 2 and 3. Moreover, the clauses of $S$ are satisfied by constraints introduced in Line 4. Thus, the oracle has an answer set

Listing 3: Finding Minimum-Size or Bounded-Size Explanations

```
{ t(A) } :- atom(A), plit(A).          { f(A) } :- atom(A), nlit(A).
#const k=0.
#minimize { 1,A: t(A), k=0; 1,A: f(A), k=0}.
:- #count { A: t(A); A: f(A)} > k, k>0.
```

Listing 4: Oracle for the Negative Case

```
{ t(A) } :- plit(A).          { f(A) } :- nlit(A).
et(A) :- t(A).
{ et(A) }:- not t(A), not f(A), atom(A).
:- clause(C), not et(P): pcond(C,P); et(N): ncond(C,N).
```

Listing 5: Extension for the Positive Case

```
simp(C) :- t(P), pcond(C,P).          simp(C) :- f(N), ncond(C,N).
simp(C) :- pcond(C,A), ncond(C,A).
:- clause(C), not simp(C), pcond(C,P), not t(P), not f(P).
:- clause(C), not simp(C), ncond(C,N), not t(N), not f(N).
```

iff $L' \cup S$ is satisfiable. However, *stable-unstable* semantics [9] and the translation *unsat2lp* from [8] yield the complementary effect, amounting to the unsatisfiability of $S|_{L'}$.

On the other hand, the positive case (the first item of Lemma 5), can be covered by extending the program of Listing 3 by further rules in Listing 5. The rules are analogous to those in Listing 1, but formulated in terms of predicates `t/1` and `f/1` rather than `plit/1` and `nlit/1`. Thus $L'$ inherits the properties of $L$, i.e., the encoding based on Listings 3 and 5 extended by facts describing $L$ and $S$ has an answer set iff $S|_{L'}$ is valid for a minimum-size $L' \subseteq L$.

## 5. Experiments

In what follows, we evaluate the computational performance of the *Clingo* system by using the encodings from Section 4 and two benchmark problems,[3] viz. the famous $n$-queens ($n$-Qs) problem and the dominating set (DS) problem of undirected graphs—recall the proof of Theorem 9 in this respect. Besides understanding the scalability of Clingo in reasoning tasks corresponding to explanation problems defined in this work, we get also some indications what kinds of explanations are obtained in practice. We study explainability in the context of these benchmark problems to be first encoded as SAT problems in *CNF* using the declarative approach from [10]: clauses involved in problem specifications are stated with rules in ASP style, but interpreted in *CNF* by using an adapter called *Satgrnd*. Thus, the *Gringo* grounder of *Clingo*

[3]https://github.com/asptools/benchmarks

Listing 6: Propositional Specification for the *n*-Queens Problem

```
1 #const n=8.
2 pair(X1,X2) :- X1=1..n, X2=X1+1..n.
3 triple(X1,X2,Y1) :- pair(X1,X2), Y1=1..n-(X2-X1).
4 queen(X,Y): Y=1..n :- X=1..n.
5 -queen(X,Y1) | -queen(X,Y2) :- X=1..n, pair(Y1,Y2).
6 -queen(X1,Y) | -queen(X2,Y) :- pair(X1,X2), Y=1..n.
7 -queen(X1,Y1) | -queen(X2,Y1+X2-X1) :- triple(X1,X2,Y1).  % X1+Y2 = X2+Y1
8 -queen(X1,Y1+X2-X1) | -queen(X2,Y1) :- triple(X1,X2,Y1).  % By symmetry
```

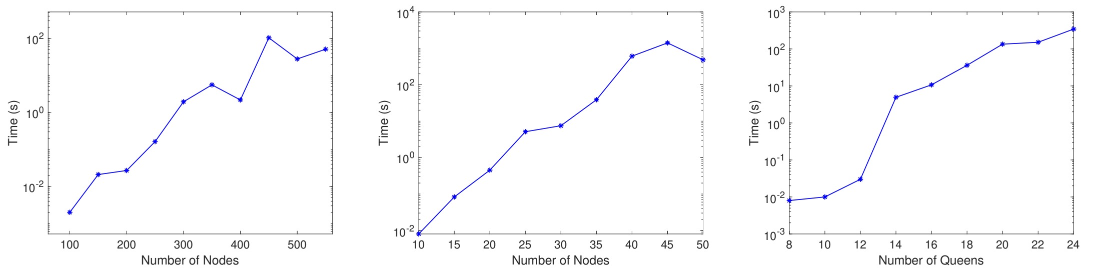
Listing 7: Propositional Specification for the Dominating Set Problem

```
1 vertex(X) :- edge(X,Y).     vertex(Y) :- edge(X,Y).
2 in(X) | in(Y): edge(X,Y) | in(Z): edge(Z,X) :- vertex(X).
```

can be readily used for the instantiation of the respective propositional schemata, as given in Listings 6 and 7, for subsequent SAT solving.

Our *n*-Qs encoding in Listing 6 introduces a default value for the number of queens `n` in Line 1 and, based on `n`, the pairs and triples of numbers relevant for the construction of clauses are formed in Lines 2 and 3. Then, for the queen in a column `X`, the length `n` clause in Line 4 chooses a row `Y` made unique for `X` by the clauses introduced in Line 5. Similarly, columns become unique by the clauses in Line 6. Finally, queens on the same diagonal are denied by clauses resulting from Lines 7 and 8. Turning our attention to the DS problem in Listing 7, vertices are extracted from edges in Line 1. The clauses generated in Line 2 essentially capture the disjunctions collected as parts of $\psi$ in (1): our encoding assumes that the edges of the graph are provided as ordered pairs for the sake of space efficiency. Intuitively, given any vertex `X` in the graph, either it is *in* the (dominating) set or any of its neighboring vertices is.

In the experiments, we evaluate the performance of the *Clasp* solver (v. 3.3.5) when used to solve various explainability problems. All test runs are executed on a cluster of Linux machines with Intel Xeon 2.40 GHz CPUs, and a memory limit of 16 GB. We report only the running times of the solver, since the implementations of grounding and translation steps are suboptimal due to our meta-level approach and such computations could also be performed off-line in general, and it is worth emphasizing that our current method has been designed to work for any *CNF* and set of literals given as input. For *n*-Qs, we generate (i) *positive* instances by searching for random solutions to the problem with different values of *n* and (ii) *negative* instances by moving, in each solution found, one randomly selected queen to a wrong row. The respective truth assignments are converted into sets of literals *L* ready for explaining. For DS, we first generate random planar graphs of varying sizes and search for minimum-size dominating sets for them. Then, *negative* instances are obtained by moving one random vertex outside each optimal set and by describing the outcomes as sets of literals *L*. For *positive* instances, we include the positive literal `in(X)` in *L* for all vertices `X`, in analogy to the reduction deployed in Theorem 9.



(a) (b) (c)

**Figure 1:** Experimental Results on Explaining DS Positively (a), Negatively (b), and $n$-Qs Negatively (c).

The results of our experiments are collected in Figure 1. Since initial screening suggests that explanation under exact size bounds is computationally more difficult, we present only results obtained by minimizing the size of $L$ using the *Clasp* solver in its *unsatisfiable core* (USC) mode. Figure 1a shows the performance of *Clasp* when searching for positive explanations for DS based on planar graphs from 100 up to 550 vertices. Explanations are minimum-size dominating sets. The performance obtained for the respective negative explanations is presented in Figure 1b. In spite of somewhat similar scaling, far smaller planar graphs with the number of vertices in the range $10 \dots 50$ can be covered. In this case, explanations consist of sets of vertices based on some vertex and its neighbors. The final plot in Figure 1c concerns $n$-Qs when $n = 8 \dots 24$ and negative explanations are sought. Explanations obtained from the runs correspond to either (i) single (misplaced) queens or (ii) pairs of (threatening) queens. The respective positive instances simply reproduce solutions and are computationally easy. Therefore, they are uninteresting.

Some observations are in order. The instances obtained by increasing their size, i.e., either the number of vertices or queens, give rise to higher running times almost systematically. For each size, the time is computed as an average for running 10 instances of equal size. Due to logarithmic scale, running times tend to scale exponentially. Moreover, positive explanations appear to be easier to find than negative ones in compliance with complexity results.

## 6. Conclusion

We have provided general, logic-based definitions of explainability and studied the particular case of propositional logic in detail. The related $\Sigma_2^P$-completeness result gives a useful, robust lower bound for a wide range of more expressive logics and future work. We have also shown NP-completeness of the explainability problems with formulas in *CNF* and *DNF* when the input truth value is restricted. Moreover, we have presented a proof-of-concept implementation for the explanation of *CNF*-formulas (without truth value restrictions). Our experimental results confirm the expected worst-case exponential runtime behavior of *Clasp*. Negative explanations have higher computational cost than positive explanations. The optimization variants of explanation problems seem interesting, because the USC strategy seems very effective and the users need not provide fixed bounds for queries in advance.

## References

[1] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103. doi:10.1145/2043174.2043195.

[2] P. Simons, I. Niemelä, T. Soininen, Extending and implementing the stable model semantics, Artif. Intell. 138 (2002) 181–234. doi:10.1016/S0004-3702(02)00187-X.

[3] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: Propositional case, Ann. Math. Artif. Intell. 15 (1995) 289–323. doi:10.1007/BF01536399.

[4] A. Shih, A. Choi, A. Darwiche, A symbolic approach to explaining Bayesian network classifiers, in: J. Lang (Ed.), IJCAI, 2018, pp. 5103–5111. doi:10.24963/ijcai.2018/708.

[5] C. Umans, The minimum equivalent DNF problem and shortest implicants, J. Comput. Syst. Sci. 63 (2001) 597–611. doi:10.1006/jcss.2001.1775.

[6] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, volume 53 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 2001. doi:10.1017/CBO9781107050884.

[7] D. Buchfuhrer, C. Umans, The complexity of Boolean formula minimization, J. Comput. Syst. Sci. 77 (2011) 142–153. doi:10.1016/j.jcss.2010.06.011.

[8] T. Janhunen, Implementing stable-unstable semantics with ASPTOOLS and clingo, in: J. Cheney, S. Perri (Eds.), PADL, 2022, pp. 135–153. doi:10.1007/978-3-030-94479-7_9.

[9] B. Bogaerts, T. Janhunen, S. Tasharrofi, Stable-unstable semantics: Beyond NP with normal logic programs, Theory Pract. Log. Program. 16 (2016) 570–586. doi:10.1017/S1471068416000387.

[10] M. Gebser, T. Janhunen, R. Kaminski, T. Schaub, S. Tasharrofi, Writing declarative specifications for clauses, in: L. Michael, A. C. Kakas (Eds.), JELIA, 2016, pp. 256–271. doi:10.1007/978-3-319-48758-8_17.

## Acknowledgments

Tomi Janhunen, Antti Kuusisto, Masood Feyzbakhsh Rankooh and Miikka Vilander were supported by the Academy of Finland consortium project *Explaining AI via Logic* (XAILOG), grant numbers 345633 (Janhunen) and 345612 (Kuusisto). Antti Kuusisto was also supported by the Academy of Finland project *Theory of computational logics*, grant numbers 324435, 328987 (to December 2021) and 352419, 352420 (from January 2022). The author names of this article have been ordered on the basis of alphabetic order.